Acquiring Planning Models from Narrative Synopses

Thomas Hayton

May, 2019

A thesis submitted in partial fulfilment of the requirements of Teesside University for the degree of Doctor of Philosophy

Abstract

The creation of planning domain models for automated planning is challenging, especially when non-technical domain experts are required for the creation of content. This is particularly true for the creation of domain models for Interactive Storytelling systems and games. AI planning can be used for the task of narrative generation and this could be utilised in such systems. Therefore a tool supported approach to the creation of narrative planning models that alleviates the requirement of specific domain modelling expertise and automates parts of the process would make narrative generation a more accessible technology in this context.

The aim of this thesis was to develop a semi-automated approach to the creation of narrative planning domain models that automates the process of domain modelling and is accessible to non-technical authors. The approach taken aimed to use narrative synopses as an input for which a planning domain model can be acquired from.

The contribution of this thesis is a novel approach for the acquisition of planning domain models from narrative synopses. The presented approach extracts the required planning information that is described by an input synopsis and from this automatically constructs a planning domain model that is representative of this information. Automated methods have been developed for the extraction of planning information that utilise having an author "in the loop" and exploit the contextual information available. A method for the automated construction of a planning domain model has been presented that is capable of reproducing the original input. This acquired planning model can then be generalised by an author using the default narrative control mechanisms that the model provides to produce a model capable of generating new story variants. The approach was implemented in a prototype system and evaluated to demonstrate the effectiveness of the approach.

Acknowledgements

Firstly I'd like to thank my supervisors: Julie Porteous and João Ferreira for their continuous support and guidance. I couldn't be more appreciative of the time and effort they've both invested into helping me throughout this process.

I'd also like to thank my previous supervisors Peter Gregory and Fred Charles for their guidance during the earlier stages of the process but also for first suggesting that I pursue a research degree.

I would like to thank Prof. Shengchao Qin and all the staff at Teesside University who have helped to support me during my time there.

I would like to thank my fellow research students and the residents of P1.07 for sharing this experience with me. I'm well aware that the constant moaning of a grumpy Yorkshireman requires a certain level of tolerance and understanding to put up with, so for that I am grateful. A special mention goes to Jamie Matthews for the sharing of countless lunch dates and being brave enough to live with me during such testing times.

Finally, I'd like to give thanks to my loving family and friends. I couldn't have done it without their ongoing encouragement and support. I promise to return the favour.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Planning is the reasoning behind action. The process of planning is the deliberation, selection and ordering of actions, based upon anticipating the outcome of each action and how this in turn changes the state of the world. Planning is a task that requires a developed understanding of the environments and objects involved. It requires an ability to rationalise over the relationships that exist within the world and quickly formulate solutions that achieve a goal, alongside an ability to adapt when confronted with unforeseen changes.

The motivation for automated planning is largely a very practical one. By creating information processing tools that provide efficient planning resources, many complex tasks such as large-scale management problems can be supported and improved by planning. There are also theoretical motivations for conducting research in the field of automated planning. Planning is an important part of rational behaviour. If one purpose of AI is to grasp the computational aspects of intelligence, then understanding planning, the reasoning behind action, is a key element of such a purpose.

1.1 Background and Motivation

In order to solve a planning problem computationally three things are required: a formal definition of the world; a definition of the problem; and a planning algorithm capable of solving it. Planning problems are modelled by separating the problem into two parts; a problem domain and a specific problem instance. The domain defines the world and the way in which it operates. The problem instance then defines the objects that exist in the world, the initial state, and the

goal criteria for that problem instance. The way in which a problem is described can have significant implications on the output plans that are produced and ultimately how suitable they will be. Given the importance of correctly modelling planning problems, the authoring of such planning models is a challenging exercise that requires time and planning domain modelling expertise.

Automated planning can be applied to a variety of tasks, one of which is the generation of narratives. Narratives can be viewed as a sequence of actions, and thus AI planning can be used to determine such sequences. AI planning has been widely used for generating narratives in Interactive Storytelling (IS) systems (e.g. Aylett *et al.* [4]; Riedl and Young [75]; Porteous *et al.* [68]). To date, the modelling of these domain models has been handled manually: a common strategy being to build up models via systematic consideration of alternatives around a baseline plot [67]. Many prototype IS systems have sought inspiration from existing narratives (e.g. Who's afraid of Virginia Woolf? [51], and Aladdin [75]).

Domain modelling for automated planning is challenging in general but is further compounded when non-technical domain experts are required for the creation of content to populate the model. This is particularly true for the creation of domain models for IS systems and games. Planning narratives for these types of application are the focus of this work; however the same is true of applications in other domains (e.g. requirements in engineering [21]). Therefore the motivation of this work is to develop a tool supported approach to help reduce the burden of creating planning domain models for narrative planning.

The work in this thesis seeks to develop a semi-automated route to authoring a baseline plot from which variants can be built: taking a single natural language plot synopsis as input and outputting a planning model. The automation of this process could potentially reduce the time and expertise required to create narrative planning models, which may in turn make AI planning a more accessible technology for content creators.

1.2 Aims and Contributions

This research aims to find a solution to the problem of automating parts of the process for the acquisition of narrative planning domain models. The approach taken in this work is that of using natural language synopses as an input source

that the required information can be acquired from. This work aims to find a general solution that can be applied to all third-person synopses and in doing so tackle the various challenging tasks that this non-trivial problem presents. One of the research questions being answered is that of: can any third-person natural language synopsis be used as a suitable basis for domain model acquisition? One of the key issues faced when learning domain models is the amount of information required. Existing approaches either require a large number of plans to learn from or complete descriptions of the state transitions that can occur. Natural language synopses represent very challenging, unconstrained input to learn domain models from, with little guaranteed regarding the information being described.

The contribution of this thesis is a novel approach to the acquisition of planning domain models from narrative synopses. The approach, referred to as StoryFramer, is comprised of a number of Natural Language Processing (NLP) and Information Extraction (IE) techniques that are used to identify the narrative information being described in the natural language synopses. Methods are presented in this work that also exploit the contextual information regarding the characters and objects that is available in this context when such information can be used to increase the accuracy with which narrative information can be identified. The approach features a method for the automated construction of narrative planning domain models that are representative of the information that has been identified in the input synopses. The planning domain models that are constructed introduce control mechanisms that allow for the regeneration of the original plot in addition to generation of new story variants.

This work refers to a creator of a narrative planning domain model as an "author". The semi-automated StoryFramer approach utilises having an author "in the loop" and the role of the author can be defined as the fulfilment of three tasks: the validation of automated processes; providing additional information; and the making of preferential choices. No requirement of planning domain modelling expertise is needed for the completion of these tasks, in keeping with the motivation to provide a tool supported approach to non-technical authors.

The contribution of this thesis is a semi-automated approach to the acquisition of planning domain models from narrative synopses. The approach features the following:

- The identification of the mentioned objects in the synopses using a combination of NLP and IE techniques and the available contextual information.

- A novel sieve-based approach to the coreference resolution of pronouns that utilises the available object information.
- The identification of the narrative events described by the synopses using NLP techniques and the obtained information. A method for the identification of objects associated with each event is presented.
- The automated construction of planning domain models that are representative of the acquired narrative information.

The StoryFramer approach is implemented in a prototype system that incorporates these features and is used to evaluate the main contribution of the thesis. Evaluations are conducted to assess the overall approach and the individual constituent components that were developed. A variety of synopses are used to test the automated components of the approach against human-identified results and alternative methods. A worked example has been provided and serves as a proof of concept, demonstrating the approach from start to finish: taking a NL synopsis and outputting a planning model capable of generating new story variants.

Our working hypothesis is that the StoryFramer approach would reduce the burden of authoring narrative planning domain models. Automating parts of the process by acquiring narrative information from synopses and constructing planning domains models representative of this information would reduce the input required of an author. A semi-automated approach would allow for the identified narrative information to be validated, ensuring the information acquired is accurate. The approach would reduce the level of planning expertise that the creation of planning domain models requires, allowing for non-experts to produce narrative planning domain models. It is also hypothesised that the available contextual information and the input of an author could be utilised in combination with NLP and IE techniques to improve the accuracy with which narrative information can be identified from natural language synopses.

The results of the evaluations presented in this thesis validate these hypotheses by demonstrating that the approach is capable of acquiring planning domain models from natural language synopses. The planning domain models that are constructed from the acquired narrative information are shown to be representative of the input synopses and capable of both: the regeneration of the original plot; and the generation of new story variants. The methods presented for the identification of narrative information are shown to achieve very positive results,

displaying significant performance increases over alternative methods through the exploitation of available contextual information and utilising an author's input.

1.3 Thesis Structure

This thesis has been organised using the following structure:

Chapter 2 - Details background information and reviews current and previous work in: Automated Planning; Planning based Narrative Generation; Information Extraction (IE) approaches for natural language; and domain authoring tools.

Chapter 3 - Discusses the key problems that have to be addressed. Presents an overview of StoryFramer, breaking the approach down into its constituent components.

Chapter 4 - Presents an in-depth look into the preprocessing of input synopses. Covering the identification of object mentions, the disambiguation and typing of objects and the coreferencing of pronouns.

Chapter 5 - Presents the acquisition of a planning domain model from preprocessed synopses. Detailing the identification of the described narrative information and the construction of planning models representative of that information.

Chapter 6 - Presents a worked example demonstrating the approach using a synopsis of The Jungle Book. Detailing every step of the approach: from input synopsis to an output planning model.

Chapter 7 - Evaluates the constituent components of the approach and the overall accuracy with which narrative information is extracted. Discusses the level of authorial input required throughout the approach.

Chapter 8 - Reviews the work presented in this thesis and summarises the contributions.

Chapter 2

Background

This chapter covers the main areas of interest for this thesis: Automated Planning, Planning Based Narrative Generation, Knowledge Engineering and Representation in planning (how the information for planning can be stored and methods of automating their creation) and extracting information from text using Natural Language Processing. Overviews of each area are presented and relevant related works are discussed.

2.1 Automated Planning

This section discusses how planning problems can be captured and represented such that they can be solved computationally. The work presented in this thesis concerns itself with the semi-automated acquisition of such representations and therefore a fundamental understanding of the process is required.

2.1.1 Automating the Planning Process

In order to solve a planning problem computationally two things are required: a formal definition of a problem and the world; and a planning algorithm capable of solving it. How the problem has been described can have implications on how suitable the final output plans will be. For example, take a classic logistics problem like that shown in figure 2.1; where by the task is that of delivering parcels around a city to a number of locations. A definition of this problem describes the starting positions of all the parcels and delivery vans. It details where each parcel is to be delivered to, but fails to capture any of the spatial relationships between the locations. While a solution can be found, the delivery route isn't

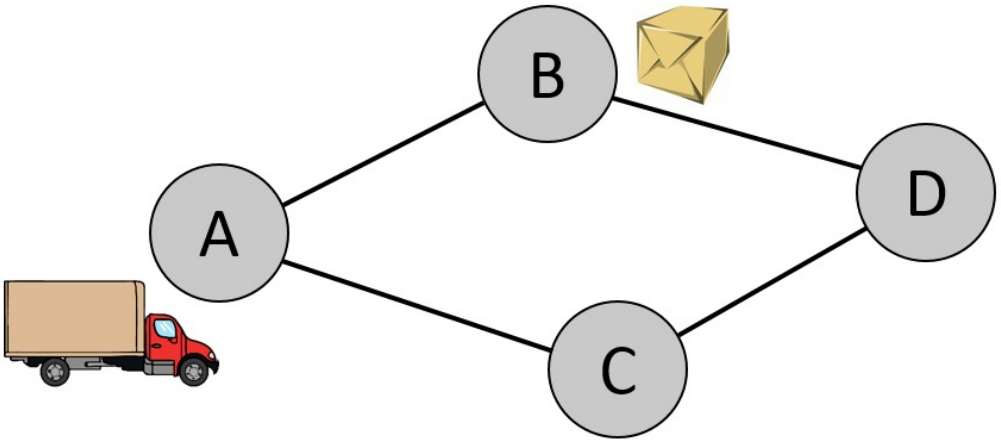

Figure 2.1: A logistics planning example: The task of planning the most efficient way to deliver parcels around a set of connected locations using delivery vans.

considered, meaning that the resulting plan might not be the most efficient solution to the given problem. This shows the importance of accurately describing planning problems. The definition and modelling of planning problems will be discussed in detail in section 2.2.

Solving a Planning Problem

Given an initial state, a goal and a set of actions that effect the world, the role of a planner is to find plans (sequences of actions) that satisfy the given problem. The task presented to planners is a search problem. The general planning problem is PSPACE-hard [11], which means that solving the problem can be as difficult as finding any other solvable solution within the same polynomial space. This means that a method of exploring all the possible combinations of actions becomes unfeasible very quickly as the space grows too large, thus research into finding more intelligent and efficient methods was required. Automated Planning has become a large field of research since Newell and Simon introduced their GPS 'General Problem Solver' [59] in 1961. The more recent introduction of the International Planning Competition (IPC) in 1998 [54] provided an incentive for the progression of Automated Planning. Since then significant advancements have been made regarding approaches to solving planning problems [39] [41] [8] and the computation of planning algorithm heuristics [38] [35] [9].

2.2 Modelling

To model a problem is to capture the dynamics of the problem space within a formal representation. Since planning is concerned with the selection and ordering of actions that change the state of the problem space, a general model for a dynamic system is required. Domain-independent (or classical) planning in the tradition of STRIPS [25], UCPOP [63], Graphplan [7], HSP [8], FF [38] and FD [35] aim to solve problems using only general algorithms rather than using domain-specific methods. Domain-specific specialised approaches certainly have their uses and are highly successful in a number of application areas where exploiting the specifics of the domain is highly advantageous; such as motion and manipulation planning in robotics [3] [46]. However, if the goal is to design an autonomous intelligent machine, limiting it's deliberative capabilities to a specific area wouldn't be satisfactory.

A domain-independent planner takes as input a problem specification alongside knowledge about the given domain. These models have to convey the actions that can occur in an abstract and general fashion. The models can vary from simplistic ones that only allow for a limited level of reasoning, to far more flexible and expansive models that can capture and reason about more complex actions. Developing formal representations for automated planning has been a area of research interest; and the problem has been modelled in a number of differing ways. STRIPS [25] was an automated planner and the same name was later used to refer to the formal language of the inputs to this planner. The STRIPS language is the base for most of the languages used for expressing automated planning problems, these languages are known as action languages. PDDL [53] is one such action language that was developed in an attempt to standardise planning languages. Newer versions of PDDL have been developed to expand the expressiveness of the language (discussed further in Sectoin 2.2.3). The method of solving a planning problem can also affect the way in which it should be modelled. Methods such as SAT [40] and SAS+ [5] cast a planning problem into an instance that can be solved as a satisfiability problem. Such approaches are often more suited towards a specific subset of problems.

2.2.1 Modelling State Transition Systems

State transition systems are a way of modelling dynamic systems. Here we define a restricted version of the model as there is no need for the consideration of external events effecting the state of the world. The applications of interest for this work are deterministic, with the world only being modified as a direct result of planned actions.

Definition 2.2.1. A state transition system is a triple $\tau = (\mathbb{S}, \mathbb{A}, \gamma)$, where:

- $\mathbb{S}$ is a finite set of states;
- $\mathbb{A}$ is a finite set of actions;
- $\gamma : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S}$ is a mapping of states and actions to states.

A state transition system is shown in Figure 2.2 continuing with the logistics example, simplified further; consisting of two locations, one delivery van and

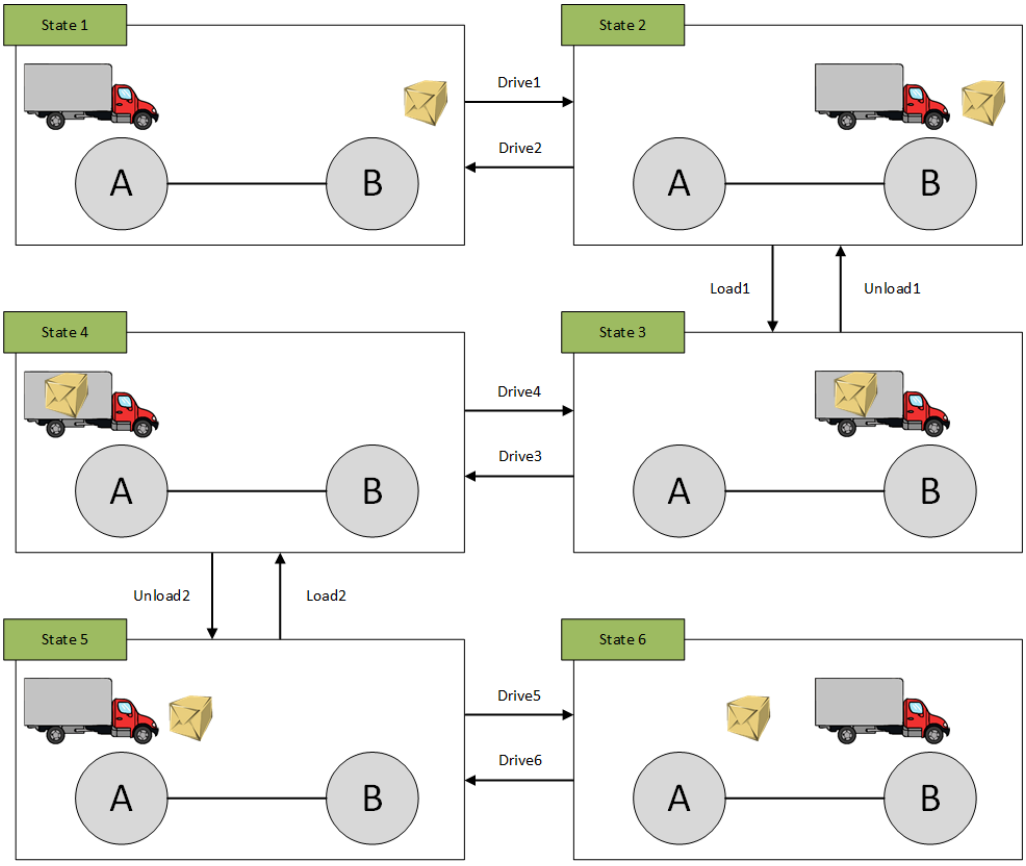

Figure 2.2: A state transition system for a simple logistics problem consisting of two locations, one delivery van and one parcel.

one parcel. The delivery van can drive between the two locations as they are connected by a road. The van can load, and unload parcels at its current location. The example shows the set of states, $\mathbb{S}$ = $\{State_1,\ldots,\ State_6\}$ and the set of actions, $\mathbb{A}$ = $\{Drive_1,\ldots,\ Drive_6,\ Load_1,\ Load_2,\ Unload_1,\ Unload_2\}$. The transition function, γ, is implied by the arcs between the states that the actions describe.

Modelling Planning Problems

Using a state transition system, τ = ($\mathbb{S}$, $\mathbb{A}$, γ), the dynamics of a planning problem environment can be captured. The actions of the planning problem are represented by $\mathbb{A}$, the possible states of the environment are represented by $\mathbb{S}$ and the state transitions of the environment are captured as γ.

The purpose of planning is to solve a specific planning problem, thus a way of defining planning problems is required.

Definition 2.2.2. A planning problem $\mathbb{P}$, can be described by the triple (τ, i, g), where:

- τ is a planning environment defined by a state transition system;
- $i \in \mathbb{S}$ is the initial state of the environment;
- g is the goal criteria that defines the objectives of the problem.

To define a planning problem for the logistics example, τ can be represented by the state transition system shown in Figure 2.2. The initial state i, could be $State_1$, with the van at location A and the parcel at location B. The objective of the problem could be to deliver the parcel to location A, meaning the goal criteria would be, parcel at location A. This problem is represented as (τ, $State_1$, parcel at location A). Using the state transition system in the figure, starting at $State_1$ and applying the following actions: $Drive_1$ in $State_1$; $Load_1$ in $State_2$; $Drive_3$ in $State_3$; and $Unload_2$ in $State_4$; the result of doing so is that $State_5$ has been reached. In $State_5$ the parcel is now at location A and the goal criteria of the problem has been met. The sequence of actions taken for this state transition to occur is a solution to the problem.

Modelling the Actions

So far the states of the world, τ have been represented using a state transition system. Each state of the planning problem can be viewed as a set of propositions; truth statements regarding the state of the world. For the logistics example, the states can be defined by where the objects (van and parcel) are located. For any given state the proposition that the Van_1 is at $Location_A$ is either true or false. A state can be uniquely identified by a set of propositions provided they have been selected well. The propositions needed to define the state for this example are as such: The van can either be ($at\ Van_1\ Location_A$), or ($at\ Van_1\ Location_B$); The parcel can either be ($at\ Parcel_1\ Location_A$), ($at\ Parcel_1\ Location_B$), or ($in\ Parcel_1\ Van_1$).

The role of the actions is to change which propositions hold true such that the new state reflects the transition that the action represents. For example the $Drive_1$ action that describes the transition between $State_1$ and $State_2$, where the proposition ($at\ Van_1\ Location_A$) holds true and ($at\ Van_1\ Location_B$) does not in $State_1$, to $State_2$, where the reverse is now true, ($at\ Van_1\ Location_A$) no longer hold true and ($at\ Van_1\ Location_B$) now does.

Actions can be modelled as add and delete rules that effect the state of the world, provided the action itself is valid and applicable in the current state. An action is represented by a triple:

- Name - Each action requires a name to act as a unique identifier. For example, the $Load$ action, ($Load\ Parcel_1\ Van_1\ LocationB$), represents the action that takes a parcel from a location and puts it inside a van.

- Preconditions - This is a set of propositions that all must hold true in the current state of the world for the action to be applicable. For example, a van can only load a parcel if they are both at the same location. The ($Load\ Parcel_1\ Van_1\ LocationB$) action is dependent on the two propositions: ($at\ Van_1\ Location_B$) and ($at\ Parcel_1\ Location_B$) both holding true.

- Effects - The effects that the action has on the world. These can be separated into add and delete effects. With add effects being propositions that are added to the state of the world and become true; and delete effects being the removal of propositions from the current state that are currently true that no longer will be as a result. The ($Load\ Parcel_1\ Van_1\ LocationB$)

action will remove the proposition ($at\ Parcel_1\ Location_B$) and add ($in\ Parcel_1\ Van_1$) from the state.

Using a PDDL representation a possible encoding of the $Drive$ and $Load$ actions are shown below in Figure 2.3.

(**:action** $Drive$
:parameters($Van_1\ Location_A\ Location_B$)
:precondition($at\ Van_1\ Location_A$)
:effect(**and** ($at\ Van_1\ Location_B$) (**not** ($at\ Van_1\ Location_B$))))

(**:action** $Load$
:parameters($Van_1\ Parcel_1\ Location_A$)
:precondition(**and** ($at\ Van_1\ Location_A$) ($at\ Parcel_1\ Location_A$))
:effect(**and** ($in\ Parcel_1\ Van_1$) (**not** ($at\ Parcel_1\ Location_A$))))

Figure 2.3: A possible PDDL representation of the $Drive$ and $Load$ actions.

The $Drive$ action moves Van_1 from $Location_A$ to $Location_B$. And the $Load$ action takes the $Parcel_1$ and puts it into the Van_1, provided they are both at $Location_A$.

Going back to the state transition system, τ = ($\mathbb{S}$, $\mathbb{A}$, γ), where $\mathbb{S}$ is a finite set of states. Using this representation means that $\mathbb{S}$ is implied by all the states that are reachable from the initial state, i, by applying any sequence of actions. This means that all the possible states of the world do not need to be declared up front; the size of the state space can grow very quickly meaning that this can be very important.

2.2.2 Modelling Planning Domains

The description of a planning problem is separated into two parts:

The Problem Domain - The domain defines the world and the way in which it operates. The Domain is a tuple, D = (O, P), that defines the set of predicates, P, and operators, O. The predicates define how the world is represented. Predicates are the relations and properties of objects that need to exist to capture the propositions of the state. In the Logistics example the predicates would be at and in. In PDDL these would be expressed as (at ?x ?y) and (in ?x ?y), to

represent that one object can be at another; and one object can be in another. The operators are a set of parameterised actions that describe the possible behaviours of the world. These actions consist of the parameterised sets of predicates that define the actions preconditions and effects, similar to the examples of Figure 2.3 which used propositions. The operators in the example would be $Drive$, $Load$ and $Unload$.

A Specific Problem - The problem model defines the objects that exist within the world, the initial state and the goal criteria. A problem model is specified with a tuple, $\mathbb{P}$ = ($\mathbb{O}$, $\mathbb{S}_i$, g), where, $\mathbb{O}$, is the set of objects; $\mathbb{S}_i$, describes the initial state, defined by the set of propositions that hold true in that state; and g being the propositions that need to hold true for the problem to be considered solved.

2.2.3 Models and Representations in this Work

The Planning Domain Definition Language (PDDL) [53] is a modelling language used in planning that has been developed for over twenty years, with PDDL3.1 [36] being the most recent version. Each version expands the modelling capabilities of the language, allowing for more planning problems to be captured. The language has become the defacto planning language with its association to the International Planning Competitions (IPCs) playing a significant part in this.

PDDL1.2 [53] was the first official version of PDDL. The model of the planning problem was separated into two parts: a domain description and a problem description. The domain description contains the elements that are present in every specific problem; and the problem description determines a specific problem. A domain and a connected problem forms a PDDL model of a planning problem that can then be used as input for a planner. In PDDL1.2 a domain can define the following: predicates, actions, an object-type hierarchy and constant objects.

PDDL2.1 [28] introduced numeric fluents, plan-metrics and continuous actions. By doing so PDDL2.1 expanded the number of problems that could be represented and solved using the language. Both derived predicates and time initial literals were both introduced in PDDL2.2 [24] to once again extend the language. PDDL3.0 [29] introduced state-trajectory constraints and preferences to enable preference-based planning. Object-fluents were introduced in PDDL3.1 [36] and

this version remains the latest and most expressive representation of PDDL.

Because PDDL has been used in conjunction with the IPCs as a means of standardising the input of planners, as a result of this many options are available for solving planning problems that have been modelled using PDDL. PDDL planning problems can often be solved using off-the-shelf planners that do not require modifying in any way. This is a significant advantage in the context of this work as the use of off-the-shelf planners requires no additional expertise from a potential user. A goal of this work is to make planning technology more accessible to non-experts and by using PDDL as the modelling language this can be achieved.

The target output used in this work is a planning problem modelled using PDDL1.2. This version of PDDL provides all of the functionality required for this purpose. The expressiveness of other versions isn't needed in this context.

2.3 Planning Based Narrative Generation

Narratives can be viewed as a sequence of actions, and thus AI planning can be used for the task of narrative generation [93]. Planning for narrative generation varies considerably compared to planning for the domains that appear in the classic benchmarks, logistics, blocksworld etc. With the IPC providing a research incentive, the goal has often been to find the optimal solution to a problem, using the quickest and most efficient methods. For a narrative plan this takes a far lower priority, with the quality of a plan being determined by other metrics, such as a plan's 'shape' or trajectory and the generation of plans that would be of the most interest to an audience. Staging an optimal plan for narrative could be considered undesirable, with a focus being placed on a plan that encounters failures and setbacks to produce a more suspenseful output. A narrative plan may also look to minimise action repetition so that an audience receives a more varied and interesting set of actions. With one application of AI narrative generation being the integration within video games and interactive storytelling, a narrative plan may need to cater for user input and external events effecting the world state, causing narratives to become invalid and in need of regeneration. Since AI planning was first proposed for the task of narrative generation by Young [93] the approach has been enthusiastically adopted and used in within various systems [13] [4] [75] [68].

2.3.1 Planning for Interactive Storytelling

Despite the planning of both traditional and narrative problems functioning in the same way, the conflict lies within differing goals for their output plans. Where traditional planning often favours optimality, the goal for interactive storytelling (IS) could vary depending on the system; to entertain or educate for example. The problem of most interest when it comes to planning for IS, is that of narrative control.

In plan-based IS systems a plan generation engine is embedded with a system where by the planner generates an initial narrative which is then presented to the user. User interaction can change the state of the planning world and hence require the replanning of the narrative. Such IS systems include the 2D and 3D visual representations of plans [51] [68], text representations [61] and filmic content [64].

To model a planning domain representative of a narrative, one approach is to start by modelling the base actions that can occur, which will form the plot. These actions should be designed in such a way that a default base storyline can be represented, while remaining generic enough that they provide enough scope for new alternative story variants.

Controlling Narratives with Constraints

In order to control a narrative plan, a method for adding the narrative information into the domain is required. PDDL3.0 [29] introduced state trajectory constraints that could be used to encode control knowledge. The PDDL modal operators "sometime" and "sometime-after" can be used to ensure the inclusion of specific key narrative events and to enforce a temporal order over them. This approach of using these state trajectory constraints for narrative control was introduced by Porteous et al. [66]. This can be done by breaking down a narrative into a set of smaller, ordered sub-goals. By breaking down the narrative into smaller sub-goals the burden on the planner is also significantly reduced. The new goal is to fulfill the criteria for the next sub-goal, reducing the time needed to generate and regenerate plans. This makes the approach suitable for interactive real-time environments, where a user can have influence over the story and require new solutions to be found.

Building on this work, these constraints were used to not only control the or-

dering of key narrative events, but also the tension level in the resulting plot [69]. Given that the domain actions had each been labelled with respective tension values, the author could specify a desired tension arc, comprising of a value to meet at each sub-goal. The planner would then select actions with the goal of matching these values as closely as possible.

Planning Believable Stories

Another metric by which narrative planning solutions are measured by is audience believability. Reidl and Young [75] describe plot believability and character believabiltiy as the two key attributes that constitute a narrative plan as being a sound and believable sequence of actions. The plan should represent a logical and causal progression of plot, and characters should be perceived by the audience as acting intentionally with motive behind their actions. There are two main approaches to planning narratives: plot-centric and character-centric. Plot centric looks at the world from the perspective of an author; having the author set up key events that need to occur during the story. This ensures plot causality but won't necessarily produce believable characters. Character-centric is the alternative approach, where characters are given their own specific goals to meet during the story. With them all working towards their own personal goals this ensures character believability. The trade off being that without higher level goals ensuring a plausible narrative, the plot coherency can suffer.

Reidl and Young developed the IPOCL planner, with the intent of striking a balance between the two approaches. Author goals and character goals were separated and both taken into consideration when planning. The main drawback of their approach was that because all goals had to be met for the plan to be valid, there wasn't the possibility for the failure of character goals; which is often a common aspect of most stories and especially crucial in comedy and tragedy [14].

Another approach to balancing plot and character was introduced by Porteous et al [67] with their Point of View (PoV) concept. Rather than characters having to meet strict goals, characters were assigned various points of view. These were characteristics such as personality traits and attitudes towards other characters. These PoVs are taken into consideration when looking at action preconditions, ensuring that characters can only take part in actions that align with

their PoV. With no set character goals to satisfy, characters can still appear to fail to achieve their personal goals and allow for more plausible story variants as a result. Building on this NetworkING [68] introduced a social network of relationships between the characters and used these attitudes generate varied and believable narratives.

2.4 Information Extraction from Natural Language

In order to be able to build domain models from story synopses, a vital part of the process is the processing of text. The difficulty of extracting the information contained in text can vary greatly depending on what restrictions, if any, the text has to adhere to. Once the text becomes more complex than very simple and specific commands, a more sophisticated approach is required in order to interpret its meaning.

Information Extraction (IE) research gained attention through the Message Understanding Conference (MUC) [33] which provided funding from the Defense Advanced Research Projects Agency (DARPA). The conference aimed to unite the efforts of IE at the time and started out with extracting the information from military messages as the goal. Since then the input has evolved, becoming increasingly more complex. As a competition, the conference provided the community with training data, test data and manually generated results that could be compared against when evaluating the system. This contributed to the development of a standard for evaluation used in IE. The common evaluation metrics used are: precision, recall and F-Measure, where F-Measure is used to combine precision and recall into a single value that can be weighted to favour either value.

2.4.1 Information Extraction Tasks

There are a number of key tasks within IE. Depending on the system and the complexity of the input texts, some tasks may not be applicable. Some of the notable and relevant tasks are:

- Word and sentence boundary identification - seemingly a fairly straightforward process, there are many edge-cases that need to be accounted for. The most common example is a full stop ".", usually denoting the end of

a sentence there are cases where this isn't the case, such as: "Dr. Watson" or "£1.50".

- Phrase identification - the identification of a group of words, or single word which plays a particular role in the grammatical structure of a sentence.
- Syntactic structure - identify the relationships between words and phrases. The result of a syntactic analysis is usually a dependency graph.
- Root word analysis - often a word has multiple forms depending on how it is being used. The most common example of this is verb tenses where the same action is being described by numerous words, such as: eat, ate, eating. Identifying the base-forms of these words to be "eat", may result in other processes and analysis easier.
- Named entity recognition - often referred to as NER, this is the task of recognising known entities within the text. NER can also refer to how these entities can then be classified into pre-defined categories such as the names of people, organisations, locations, etc.
- Relationship extraction - the identification of relations between entities.
- Coreference resolution - the task of finding all expressions that refer to the same entity in a text. The most common case where coreference resolution is required is when pronouns are present in the text. The process of coreferencing aims to establish who or what a pronoun is referencing. An example being, "Bob overslept and he was now running late for work." Here it is important to know that 'he' is referencing 'Bob' if the information is to be extracted correctly.

The ordering of these tasks is important as many are dependent on other tasks having been already completed. For example, named entities have to first be identified before an attempt can be made at extracting relationships and links between them.

For many IE tasks 'perfect' solutions do not yet exist, with ongoing research aiming to increase the accuracy of specific tasks or general and more tailored inputs. NLP research is often specific to a given language. The more structured a language's grammar the easier it is to extract information from. For this work

and any of work referenced, the language being used is English. English is also the dominant language in research and on the WWW with approximately 52% of websites using it for their content [90].

Natural Language Processing Toolkits

Natural language processing toolkits provide NLP functions and annotations that can be used and applied in information extraction systems. Such annotations include methods for: lexical analysis; text chunking; part-of-speech tagging; named entity recognition; coreference resolution; constituency parsing; dependency parsing; and sentiment analysis.

The available NLP toolkits include Stanford CoreNLP [48], NLTK [45] and Apache OpenNLP [60]. The table below summarises the features of each toolkit.

	CoreNLP	NLTK	OpenNLP
Lexical Analysis	✓	✓	✓
Text Chunking	✓	✓	✓
POS tagging	✓	✓	✓
Named Entity Recognition	✓	✓	✓
Dependency Parsing	✓	✓	✓
Constituency Parsing	✓	✓	✓
Coreference Resolution	✓		✓
Sentiment Analysis	✓		

Table 2.1: A summary of the features provided by NLP toolkits.

Stanford CoreNLP is used for this work as it includes all of the functions and annotations that are required. CoreNLP is also the most well documented of these toolkits.

2.4.2 Named Entity Recognition

Named entity recognition is the task of locating and classifying named entities in a text into pre-defined categories such as persons, locations, organisations, expressions of time, quantities, etc.

NER is treated as two distinct problems: the detection of entities, and their type classification. The first phase is typically simplified to a segmentation prob-

lem similar to chunking; which is the task of parsing natural language sentences into partial syntactic structures. Entity names are defined to be a contiguous span of tokens, with no nesting, so that "The Queen of England" is a single name, disregarding the fact that inside this name, the substring "England" is a name itself. The second phase of classification requires choosing an ontology that machine learning models can be trained on so that an attempt at categorising the entities by type can be made.

Approaches to Named Entity Recognition

The two NER problems of detection and classification are both considered to be difficult non-trivial problems that are yet to be solved, with the highest F-Score achieved at the CoNLL03 competition [84] being 88.76.

Approaches typically use a combination of linguistic grammar-based techniques alongside statistical models such as machine learning. Different statistical models have been used for NER, such as the Hidden Markov Model (HMM) based chunk tagger [95] or the approach adopted by Stanford's CoreNLP NER [26], which instead utilises conditional random fields (CRF).

The overall systems used for many language technology applications tend to run several independent processors over data. Such processors include parsers, named entity recognisers and coreference systems. Such approaches can easily result in inconsistent annotations which are harmful to the performance of the aggregate system. The system proposed by Finkel & Manning in [27] joined both the parsing and NER phases together, improving the performance of both tasks in doing so. A joint, discriminative model is used, which is a feature-based CRF-CFG (Conditional Random Fields/Context Free Grammar) parser that operates over tree structures augmented with NER information.

Statistical supervised training NER systems typically require a large amount of manually annotated training data. One of the main current research efforts in the field is to reduce the annotation labour required. Employing a semi-supervised approach like that of Lin & Wu's phrase clustering [43] is one way of doing this. This method uses supervised training to learn word clusters but then goes a step further and utilises a semi-supervised algorithm to also use phrase clusters as features. Out of context, natural language words are often ambiguous. Phrases are much less so because the words in a phrase provide con-

text for one another. By using phrase clusters an improved F-Score of 90.9 was achieved on the CoNLL03 evaluation data. Alongside such approaches, projects like that of Zhai et. al [94] have turned to crowdsourcing as a means of obtaining high-quality aggregate human judgements for supervised and semi-supervised machine learning NER systems.

Even state-of-the-art NER systems are brittle, meaning that NER systems developed for one domain do not typically perform well on other domains [65]. Considerable effort is involved in tuning NER systems to perform well on new domains; this is true for both trained statistical and rule-based systems. Early work in NER systems was aimed primarily at information extraction from journalistic articles before then focusing on the processing of military dispatches and reports. Both the CoNLL03 and MUC7 are subsets of news corpora and still remain as two of the standard evaluation datasets. A main focus of the field described by Ratinov & Roth [74] is to achieve robust performance across domains and overcome the difficulties involved. The automatic content extraction (ACE) program presented in [22] included several types of informal text styles, providing more variety than it's predecessor MUC. More recently NER has been attempted on more noisy, informal text in the form of tweets by Ritter et al. [76], standard NLP tools saw severely degraded performance on such input.

With there being no perfect solution or general robust NER system available at present integrating NER into a new system would required either a new approach that exploits the specifics of its intended domain or substantial relevant training data for which to train an existing model on. Extensive manual human annotation of such data would then need to be undertaken before it can be used for training.

2.4.3 Coreference Resolution

Coreference resolution is the detection of coreference and anaphoric links between entities. In IE this is typically restricted to finding links between previously extracted named entities. If an entity hasn't been identified during the NER phase of processing, no links can be extracted related to that entity either. When solutions for coreference resolution are evaluated, they are tested on a sets of evaluation data provided by a conference such as the SIGNLL Conference on Computational Natural Language Learning (CoNLL) [71]. The evaluation data

contains both nominal and pronominal references. All the entities are manually pre-identified so that only the coreferencing component is being evaluated. This does mean that when applied to real-world problems the accuracy scores may suffer due to the failure of previous processes that the coreferencing is reliant upon.

Focusing on the Stanford CoreNLP toolkit [48], there are three coreferencing solutions available to choose from: deterministic, statistical and neural. These approaches differ in speed, accuracy and the amount of model training required.

Stanford Coreference Resolution: Deterministic Approach

The Stanford Deterministic Coreference Resolution System is a fast, rule-based approach to coreference resolution. The method described by Raghunathan [73] utilises a multi-pass sieve approach to the problem. For a given mention a list of candidate antecedents is created by selecting preceding mentions. For the same sentence, Left-to-right breadth first traversal of the syntactic tree is used for this as favouring subjects closer to the beginning of the sentence yields more probably candidates. This is also the case when looking to previous sentences for a pronominal mention as subjects are more probable antecedents for pronouns [42]. When looking to previous sentences for a nominal mention however, right-to-left breadth first traversal is used, instead favouring document proximity.

Once a list of candidate antecedents has been compiled, they are passed through several sieves that looks to allocate the mention to a coreference cluster with other mentions that reference the same entity. The sieves are a selection of rules which determine whether a link has been found between entities. The first pass starts with very strict constraints, initially only looking for exact matches, and gradually they become more relaxed. The final pass is dedicated to the coreferencing of pronouns.

Pronominal coreference resolution is handled in this method by enforcing agreement constraints between coreferent mentions. The attributes used for these constraints are: number, gender, person, animacy and NER label. These are added to mentions based upon static lists and dictionaries. If values cannot be detected the attributes are set as unknown and are treated as wildcards, i.e., they can match any other value.

Stanford Coreference Resolution: Statistical and Neural Approaches

Both the statistical [15] and neural [16] approaches share a similar method and mainly differ in the models that are trained and used. The statistical approach uses machine-learning and for neural, neural-networks are utilised. Both focus on the training of pairwise mention-ranking models. Given a mention m and a candidate antecedent c, the mention-ranking model produces a score, $s(c, m)$, indicating their compatibility for coreference. The score is based upon a variety of common features, for the full set of features see [17]. Some of the features include:

- Distance - the distance between the two mentions in sentences; number of mentions between them.
- Syntactic - POS (part-of-speech) tags of the first, last and head word; number of noun phrases under a mention on the syntactic tree.
- Semantic - named entity types; speaker identification.
- Rule-based - exact and partial string matching.
- Lexical - the first, last and head word of the mention.

The approaches differ in the models and the training algorithms that are used for this task. These models have to be trained using a training set of data. The accuracy of these models will improve given a larger amount of training data. In this case training data is provided alongside the competition data sets that the models were evaluated on. There's no guarantee that models trained on such data would be as effective when applied to real-world applications.

The statistical approach also looks to make use of entity-level information by combining this pairwise ranking with an entity-centric coreference solution. In this case the pairwise ranking only looks to find a single best antecedent for a mention. Mention pairs alone however are not enough to produce a final set of coreference clusters as they do not enforce transitivity. A pair of mentions (a, b) and the pair (b, c) are both deemed coreferent by the model, but there is no guarantee that (a, c) will also classify as coreferent. In order to cluster these mention pairs into larger clusters, entity level information is used. Attributes gender and NER labels are identified and assigned to the entities and used to

constrain which clusters can be merged. The attributes must be shared across the entire cluster. The decision of whether two clusters should be merged is controlled by a trained ”agent” that uses an agglomerative clustering approach. More different training data is required to effectively train this agent model.

Stanford Coreference Resolution: Evaluation of approaches

Table 2.2 below shows the performance results for the three coreference systems that Stanford CoreNLP provides. These results are taken from their website [82], where additional information about each system can be found.

	Preprocessing Time	Coref Time	Total Time	F1 Score
Deterministic	3.87	0.11	3.98	49.5
Statistical	0.48	1.23	1.71	56.2
Neural	3.22	4.96	8.18	60.0

Table 2.2: The performance results of the different coreference solutions provided by the Stanford CoreNLP toolkit.

The F1 scores are the scores of each system run on the CoNLL-2012 [70] english evaluation data. These models are designed for general-purpose use so these numbers are lower than those reported in each associated paper. The preprocessing time represents the time required for POS tagging, syntax parsing, mention detection, etc., while coref time refers to the time taken by the coreference system.

The results show the varying accuracy and speed of each solution, with the neural approach proving to be the most accurate but also the slowest. Statistical is the fastest method based on total time due to only requiring dependency parses, rather than the far slower to produce constituency parses. The main advantage of deterministic is that as a fast rule-based approach there is no need for any models to be trained beforehand. These results show an overall coreference score that includes both nominal and pronominal entity referencing. 38.69% of the mentions in the CoNLL-2012 test data were pronouns. There is no requirement to publish the results of how each system fared on specific categories of mentions. Results may vary significantly depending on the application for which the coreference systems are being used. If being used to only coreference the pronouns in a text, no assumption can be made that these values would hold true. With that being

the case the general coreferencing methods provided with the CoreNLP toolkit aren't adequate in the context of this work.

2.5 Domain Model Acquisition

AI planning has been shown to be a useful approach for the generation of narrative in multimedia storytelling systems. However the creation of the underlying domain models is challenging: the well documented modelling for AI planning authorial bottleneck is further compounded by the tendency for authors to be non-technical. Formulating and maintaining domain models is considered a central challenge in knowledge engineering for AI planning; in particular overcoming the need to hard-code and manually maintain such models. The field of domain model acquisition aims to explore alternative methods for obtaining/learning planning domain models, either fully or partially.

2.5.1 Learning Domain Models from Sets of Example Plans

One method of learning domain models is to use sets of example plans, also referred to as plan traces, as it is possible to automatically infer the underlying transition system from them. Using such an approach removes the necessity for the domain expert to also be an expert at modelling transition systems.

The LOCM system [19] learns planning from sets of example plans. Its distinguishing feature is that the domain models are learned with no observation of the states in the plan or of predicates used to describe them. LOCM exploits the assumption that an action will change the state of the objects involved in some way, and that the objects need to be in a certain state before the action can be executed. By tracking all the individual objects involved in a plan trace it is possible to work out the underlying state machines. For example, in a logistics delivery domain it is possible to observe that once a parcel has been loaded into a van, it can't be loaded into another van, without it first being unloaded at a location. The system works because of some restrictive assumptions regarding the form of the model. Objects are grouped into sorts and the behaviour available to objects of any given sort is described by a single parameterised state machine. LOCM is very powerful at finding satisfactory models that fit within this representation, however for many domains, including many of the benchmarks, this model is too restrictive to sufficiently capture the domain's semantics. The expressiveness of

LOCM is more restrictive than STRIPS.

LOCM2 [18] built upon LOCM by weakening some of the assumptions to allow for a more expressive representation, enabling a wider range of domains to be captured and learnt. LOCM2 allows for separate aspects of an objects behaviour to be represented as separate state machines. The LOP algorithm [31] goes a step further, inducing static predicates by using a combination of the output from LOCM2 and a set of optimal plans as input. LOP works by finding the minimal static predicate for each operator that preserves the original length of the optimal plan.

NLOCM [32] implemented an approach for learning numeric domain models with fixed action costs. Still using a set of example plans as input, the only additional information required is a overall plan cost associated with each plan. Using a constraint programming approach the cost of each individual action can be calculated, given a sufficient number of input plans. NLOCM-BF [34] relaxed the approach, allowing for a best-fit estimation of a domain model's action costs instead. This was then applied to the narrative application of learning tension values for each action of a domain model that represented a number of episodes from a cartoon.

Aineto et al. [2] present an approach for learning STRIPS action models from examples that is flexible in the amount the amount of input knowledge available, accepting partially specified models as input. The approach is can be applied to input ranging from sets of example plans to just a pair of initial and final states where no intermediate states or actions have been observed.

2.5.2 Framer: Learning Domain Models from NL Descriptions of Plans

The Framer system [44] presented by Lindsay et al. implements an approach for learning planning domain models directly from natural language (NL) descriptions of activity sequences. For modelling tools, there is still the underlying assumption that the user can formulate the problem using some formal language. Even in the case of using domain model acquisition tools such as LOCM [19], there is still a requirement to specify input plans in an easily machine readable format. Providing such input is impractical for many potential users and thus motivated an approach for learning domain models directly from natural language.

Starting with natural language descriptions of actions, the goal of Framer is to construct formal representations of the action sequences from these using NL analysis. The generated action sequences provide the necessary structured input required to inducing a PDDL domain, using existing domain model acquisition technology, in this case LOCM.

The first step in the approach is to generate action templates: reduced representations of the input sentences, capturing the main action, objects that are mentioned and an indication of their roles in the sentence. This is done by utilising Stanford's CoreNLP toolkit, specifically the dependency graph output. Taking the dependency graph annotation, this representation is further simplified to move closer to a predicate logic representation. This is achieved by applying a recursive set of rules that crawl the dependency graph, transforming the relations based upon their type. The root verb of the sentence forms the basis of the action name, while the verbs subject and objects form the arguments. Conjunctions introduce new sentence clauses, which form further predicates. Modifiers and compounds are other relation types of interest, as these are used to transform the names of the predicates and arguments.

Once an action template representation has been generated, an attempt is made at clustering together sentences that describe similar behaviours using online lexical resources and a similarity metric. It is at this point that a consistent formulation of the input sentences is generated, which is sufficient enough for a domain model to be induced from.

In their evaluation Lindsay et al. [44] demonstrated that, with certain restrictions on the NL input, that it is possible to learn planning domain models and generate PDDL representations from natural language input. The approach relies on consistent object referencing being used throughout and requires actions to be fully described. Action descriptions will often lack key information. For example, "Drive the Truck to Location A" would be an insufficient action description as it neglects to mention where the Truck is driving to Location A from. Once a user understands the restrictions on the input sentences, the need for any specialist modelling knowledge has been alleviated by this approach.

There are significant obstacles preventing such an approach being used for the purpose of learning domain models from narrative plot synopses: 1) Unless the synopsis is written in such a way to adhere to the input constraints, sourcing a synopsis online that meets this criteria isn't feasible due to how specific the

requirements over the natural language are. 2) A synopsis effectively describes one input plan and for Framer to be able to accurately induce a model, a large set of plans is required.

2.5.3 Extracting Planning Information from Natural Language

An alternative method of domain model acquisition is to extract planning information from natural language textual input, approaching it as an information extraction task. A system doesn't have to be explicitly extracting information for the purpose of creating a domain model for it to be classed as extracting information relevant towards planning. Depending on the input text being used, it is very likely that extracting all the possible information available from the text won't be enough on it's own to fully learn a domain model from [44]. With that said it is still important to extract all the planning information present, from which a planning model can then be built upon.

The purpose of a domain model is to capture the semantics of a problem space. Any information relating to the world and the state transitions that can occur is of relevance and should be extracted. What exactly is dependant on the type of input being used and may also be dictated by the nature of the domain being described. Information that is likely relevant across all domains would include:

- Actions/Events - Identifying when actions and events have occurred or are being described. Additionally being able to associate which objects are involved in a given action, and if mentioned the possible preconditions and effects for an action/event.
- Objects - Identifying all the objects mentioned in the text. This includes both named entity recognition and coreference resolution so that every mention is being associated to a uniquely identifiable object.
- Predicates - Extracting the properties of objects in relation to the world and potentially relations with other objects.

Information extraction tasks are usually divided into separate tasks of identification and classification, and this holds true when applied to extracting planning information. Identifying objects from natural language and knowing what type of objects they are are two separate tasks.

Extracting Planning Information from Instructions

Instructions can be viewed as detailed information about how something should be done or operated. They can also be seen as a description of a plan that solves a specific problem, making them an ideal input to learn domain models from. Crucially, instructional language is situated: it assumes a situational context which the agent (i.e. the reader) is to carry out a sequence of actions, as applied to objects that are (or become) available in the immediate environment. Instructions are usually written in imperative form, have a simple sentence structure and are highly organised. While instructions do appear to make for ideal natural language input, there are still a number of challenges that need to be addressed. The main drawback is that actions and objects may not be explicitly specified; it's common for instructional language to be ambiguous, under-specified and often even ungrammatical compared to more conventional usage.

Malmaud et al. [47] used cooking recipe instructions as an example for the automated interpretation of how-to instructions. This approach interprets entire recipes as opposed to independently parsing each sentence. The model used has the overall structure of a discrete-time, partially observed, object-oriented Markov Decision Process, with the goal of inferring the recipe, given the noisy evidence. The paper discusses numerous challenges that were faced regarding the interpretation of arguments, actions and control structure:

- Arguments - Recipes often feature arguments that are omitted or dependent on the context. Common ingredients such as water, ice, salt and pepper often aren't listed on the corresponding ingredients list and are assumed to be available. In some case arguments are never mentioned and are the result of a previous stage. In the example "Pour ingredients over ice and shake vigorously," the object to shake is the unmentioned container that the other ingredients are now in. Given the instruction to "Add the remaining ingredients," requires an understanding of which ingredients have already been used prior in recipe.

- Actions - Actions can also have ambiguous meanings and can be omitted from recipes and simply assumed. For example most recipes involving eggs will instruct the user to "Add eggs to the mixture," with the need to crack them being implied but not mentioned. Actions such as "garnish" can be referring to numerous smaller processes assumed from context or

obtained from a visual representation of the final product that will often accompany cooking recipes.

- Control Structure - Many conditions mentioned require additional interpretation. Often the duration of an action is specified by a final condition, such as "Whisk until soft peaks have formed," and this requires knowledge of an expected state. Additionally some instructions are left open to preference, "Add salt to taste", for example. Though implicitly sequential, recipes can also include sequencing language. Certain steps may require a user to complete "ahead of time", or to be completed while waiting for another step that is currently in progress.

Addis & Borrajo [1] look to extract plan descriptions from semi-structured online documents such as those available at www.WikiHow.com. Here the template nature of these electronic documents is exploited to crawl the web pages and process the relevant information. The ingredients/equipment list is extracted from the page alongside the raw natural language method steps. An extracted plan is then built up by taking the objects required to form the initial state. Semantic analysis of the method sentences is conducted using WordNet [55], so that the relevant information, such as the verbs conveying actions can be identified. These actions and the objects associated to them form the extracted plan, with the title of the article, "how-to-do-x" becoming the goal. The presented system achieved a accuracy of 68% across 40 random articles. Overall the extracted plans were deemed to be understandable by a number users, though clearly far from perfect. Despite adhering to a set template on WikiHow, articles are still authored by many people, communicating with differing writing styles and levels of detail. The approach claimed to have reached their goal of having a good trade-off between information retrieved from the web pages and the information lost during the analysis and filtration process.

Yordanova [92] proposes an information extraction approach to learning precondition and effect rules that describe user behaviour from textual instructions for activity recognition. As with previous methods, NLP techniques are used to extract the semantics of the input. An attempt at identifying causal relations between the actions is made by transforming every word of interest into a time series and performing time series analysis to identify any causal relations. This does however rely on the same actions being repeated multiple times during a

plan in order for such relations to be detected.

Instructions can also be in the form of spoken input. Both Thomason [83] and Scheutz [77] present methods that look to extract and learn information from spoken instructions and commands in the context of human-robot dialog. In this environment the goal is to identify the actions and objects involved from spoken NL sentences. Spoken sentences are converted into text such that semantic and syntactic language analysis can then be used to extract the information required.

Extracting Planning Information from Narrative Text

Narratives can be viewed as a description of characters and the events that they are involved in. Such information could be extracted from a narrative to form part of a narrative planning model.

The automatic extraction of such information is an area of interest in research, with the manual extraction and representation of such information viewed as an authorial bottleneck that could be alleviated. Similar to using textual instructions the goal of the task is to identify all the information regarding the actions, arguments, relations, control structure, and any other criteria that may be of interest. Narrative text however is a far noisier input in comparison, proving to be challenging to learn from. Narrative descriptions or synopses are mainly written in third-person with the inclusion of pronouns and multiple references to the same objects; and often encompass a wider range of writing styles and the level of detail being described can vary significantly.

Goyal et al. [30] attempt to automatically extract narrative information from narrative text in order to produce plot unit representations. They focused on identifying the characters involved in a story, as well as the affect states for each character as a result of the actions they are involved in. Plot units include three types of affect states: positive, negative and mental. The idea being to identify if the result of an action is desirable, undesirable or introduces a motivation for a given character. In the example, “The cat ate the mouse,” the action would yield a positive state for the cat who has been fed, and a negative state for the mouse who has been eaten. In order to do this a lexicon of patient polarity verbs that impart a positive or negative state on their patients was manually created as such knowledge is not readily available on existing semantic resources. Aesop’s fables were used as the input for this work. With the extraction of plot units from

narrative text being a hugely complex task, no gold-standard evaluation data exists. Fables represent relatively simple and constrained texts. The problem of identifying characters and resolving pronouns was minimised with some simplifying restrictions put on the input data: only two characters were allowed per fable and both characters are mentioned in the fable's title. This allows for a process of elimination method to be used for coreference resolution, where the identification of one of the character's genders using a simple rule-based coreference system can lead to the identification of the other character based on the types of pronouns being used in the text.

Rather than focusing on learning the sequence of events by which a narrative is defined, alternative methods focus on the identification of personas and the roles of the characters that drive these narratives. Valls-Vargas et al. [85] refer to Propp's Morophology of the Folktale [72] which categorised narrative roles that characters can play into (Hero, Villain, Dispatcher, Donor, Helper, Prize, and False hero) and hypothesised that the information given about how the characters behave towards one another can help identify their roles. Actions that are likely to occur between two specific roles of character were encoded into a matrix. For example, a fight is likely to occur between a Hero and a Villain. Every character in a narrative is then assigned a role and a score is computed based upon how well their interactions fit the matrix.

The approach taken by Bamman et al. [6] attempts to learn the personas of characters in film. These lexical classes capture the stereotypical actions of a character, as well as attributes by which they are described. Personas are defined by three typed distributions: words for which the character is the agent, words for which the character is the patient, and words which are character attributes. They used the example persona of a *zombie*, that could be categorised as an agent that kills and eats, is killed by other characters, and has the attribute of being dead. Given this information models are then trained on data using an unsupervised learning algorithm before then being used to identify these personas in movie plot summaries taken from Wikipedia.

2.6 Domain Authoring Tools

As discussed in the introduction, the task of authoring planning domains is complex, time consuming and requires expertise of planning domain modelling.

There is a motivation for tool supported domain modelling approaches that help to simplify, automated and make AI planning more accessible to non-technical authors. This section reviews a number of the available domain authoring tools available.

2.6.1 itSIMPLE

itSIMPLE [86–89] is an IDE for modelling planning domains. The tool was designed to support users during the construction of a planning domain application mainly in the initial stages of the design life cycle. The stages include processes such as domain specification, modelling, analysis, model testing and maintenance, all of which are crucial to the success of the application.

itSIMPLE started by basing itself around the semi-formal specification language, UML [23], which is a well-known diagrammatic language. With the tool being aimed at both those who are familiar and unfamiliar with AI planning using a more general specification language keeps the barrier to entry minimal. The environment also utilises XML [10], Petri Nets [57] and PDDL [53] representations, each one contributing towards the whole design process. Petri Nets are used for dynamic domain analysis and PDDL is used based on the AI planning communities adoption of the language as the standard specification for planner inputs.

Using a specialised use of UML to model planning domains is what makes itSIMPLE unique. The UML editor provided visualises the domain modelling process using 5 diagram types: use case diagrams; class diagrams; state machine diagrams; timing diagrams; and problem instance diagrams.

itSIMPLE provides a visual approach of diagrammatically modelling planning domains based upon UML. The tool can translate the model into a PDDL representation which alleviates any required knowledge of the language, that many non-planning users are unlikely to have encountered before. A PDDL editor is provided for those with a working knowledge of the language. Also provided are a number of planning algorithms built in to the tool, alongside methods of analysing the created models.

In order to correctly produce a working domain model using itSIMPLE a user is still required to have a good understanding of how planning domain models operate and function, and how to represent this in the non-standard form expected

by the tool. On top of this the definition of action preconditions and effects is done using a special UML language, the Object Constraint Language (OCL). Again this requires the user to understand what input is expected from them and the correct syntax in each case. It could be argued that despite alleviating the need to have some expertise in how to write domain models in PDDL, this has just been replaced by the need for a more specific, non-standard knowledge required to successfully produce models using itSIMPLE.

itSIMPLE wouldn't be applicable in the context of this work as the tool isn't accessible to non-technical authors. The tool also provides no means of automating any of the domain modelling process.

2.6.2 GIPO

GIPO [52, 79–81] (Graphical Interface for Planning with Objects) was built to investigate and support the knowledge engineering process in the building of applied AI planning systems. The research was primarily directed at tackling the end-user problem for the engineer who might be a domain expert but won't necessarily have a specialist knowledge of AI planning.

GIPO embodies an object centred approach and provides a graphical means of defining a planning domain model. Crucially tools for importing and exporting domain definitions to a literal-based PDDL format are included, in-keeping with the goal of supporting a user without such expertise. The tool also makes available a range of validation tools to perform syntactic and semantic checks on the domain model, alongside a number of third party planning algorithms that can be run from inside the tools environment.

Although using an object centred graphical method lifts the process of planning domain modelling to a conceptual level, the details of specifying state transition systems are still too theoretical for an unskilled user. GIPO does however successfully remove the need for knowledge of a specific domain modelling representation such as PDDL and by favouring a graphical approach goes some way to visualising the modelling task. For those with modelling expertise, the built-in model analysis tools, planning algorithms available, and the ability to import PDDL does give the tool some utility and appeal.

GIPO also wouldn't be useful in the context of this work as there is no attempt to automated the domain modelling process. Authors still require domain

modelling expertise as the models are manually created using the GIPO tool.

2.6.3 Planning.Domains

The Planning.Domains platform [56] has been described as an initiative with the fundamental objective of providing a set of resources, repositories and tools for researchers to discover and develop planning problems. Planning.Domains consists of three primary components: 1) An API for existing planning problems and benchmarks; 2) The Planning.Domains Solver - an open and extendable interface to a planner in the cloud service; and 3) a fully featured online editor for creating and modifying PDDL.

The PDDL editor includes a number of standard features (e.g., syntax highlighting, bracket matching, and code folding), alongside other custom features: 1) PDDL specific auto-completion; 2) Integration with both the API and Solver to allow for the importing of a wide variety of domains and problem files, as well as the ability to compute and display solutions during the editing process; 3) Problem analysis can be conducted using an online version of TorchLight [37]. Planning.Domains also features a plugin framework that allows users to expand the tools capabilities themselves.

Planning.Domains succeeds in providing an easy to use online PDDL editor. With the tool being aimed at researchers and those already with an expertise of PDDL and planning, no attempt at making planning technology more accessible to a wider audience has been made. Planning.Domains supports the manual authoring of a PDDL planning model. One of the aims of this work is to automate the domain modelling process, which Planning.Domains doesn't achieve.

2.7 Conclusions

This chapter provided an overview of the areas of interest for this work. Automated planning and the modelling of planning problems have been reviewed in addition to ways in which automated planning can be applied to the application of narrative generation. Approaches to domain model acquisition have been discussed, including how information extraction techniques can be used to to extract planning information from a variety of natural language sources. Domain authoring tools have been reviewed and their key features identified.

Approaches to information extraction are often focused towards general solutions. When these approaches are applied within a specific context, such as narrative synopses, information that is available can often go unused. By exploiting the additional information available within a given context, information can be extracted with greater accuracy.

The existing domain authoring tools that are available are aimed towards authors with a knowledge of modelling planning domains. These tools do little to alleviate the required expertise, making such tools unsuitable for non-technical authors. By automating the construction of planning domain models, narrative generation technology can be made more accessible to those who may utilise it within games and IS applications.

Chapter 3

StoryFramer: Acquiring Planning Models from Narrative Synopses

The contribution to knowledge presented in this thesis is a novel approach for the acquisition of planning domain models from narrative synopsis. StoryFramer is the name used to refer to this approach throughout this work. The goal of StoryFramer is to take a natural language description of a story (plot synopsis), and from this create a planning domain model which could be used with a planner.

This chapter will introduce an example narrative synopsis that will be used throughout this work for the purposes of illustrating various aspects of the StoryFramer approach. A problem description is formulated based upon the nature of the input, highlighting the key problems that a solution to this task has to address. An overview of StoryFramer is presented, breaking the approach down into its constituent components. This chapter will help demonstrate the goals and contributions of this work. Following chapters will review StoryFramer's individual components in greater detail.

3.1 Scooby-Doo: An Example Input Synopsis

Using natural language descriptions of stories as input presents numerous challenges when attempting to extract information from them. Synopses are more than just a simple plot outline: a synopsis is a miniature story, with actions, descriptions, characterisation and snippets of dialogue to emphasise the narrative being told. To illustrate the problems that are faced, an example input synopsis, a section of which is shown in Figure 3.1, will be used throughout this work.

The gang travels to the museum (now the next day) where they deliver the knight to the museum curator, Mr. Wickles. He thanks them, but fears that perhaps it wasn't a good idea with Professor Hyde White disappearing. He goes on to explain about the legend of the Black Knight and how it comes to life when the moon is full. Velma asks him what Professor Hyde White was doing with it (despite having already found out beforehand), and replies that the professor was delivering it to the museum all the way from England. As this is going on, they don't notice the knight's glowing eyes. Two workers begin to move the crate, one of them asking Mr. Wickles where to put it. He tells them to put it in the medieval room. As Scooby follows the workers, he finds a strange pair of glasses. He picks them up, as Daphne calls him, while Fred says they're leaving.

While driving down town, Velma says that the mystery has her baffled, and has got Shaggy hungry, asking when they can eat? Scooby pops his head up, in agreement, while still wearing the glasses he found. The others notice, and realize he must have found them at the museum. Shaggy wonders what they're for, with Fred suggesting they go to the library to find out.

At the library they read a book which says that the glasses are for jewelers, scientists, and archaeologists like Professor Hyde White. It also says they're made in England. These two clues indicate that something is definitely up, and the gang plan on returning to the museum to investigate.

The gang return to the museum at night and break in through an upper window. They split up and look for clues, not knowing that spooky eyes in an Indian effigy are watching them. Scooby, Shaggy and Velma bump into the Black Knight and have a brief altercation. Scooby runs into the fossil exhibit and begins gnawing on the bone, but is chased by the Knight. He meets up with Shaggy and the two find one of the paintings is missing. He informs the gang, but when they return, the painting is back on the wall.

Figure 3.1: A section taken from the example Scooby-Doo input synopsis [78]

The synopsis describes the plot of ‘What a Night for a Knight’, which is the first episode of the well-known cartoon, ‘Scooby-Doo, Where Are You!’ [78]. This synopsis provides a good example of StoryFramer’s target input: a synopsis written by a human, sourced from the internet. The only restrictions placed on the input synopsis are that it has to be written in English, and in third-person. It also contains examples of the common problems regarding the input that this task is required to deal with, all of which will be highlighted in the following section.

3.2 Problem Description

Acquiring a planning domain model from an input natural language synopsis is a complex task that can be decomposed into two main stages: preprocessing the input synopsis; and domain model acquisition. In the following sections the problems these tasks pose are illustrated in reference to the Scooby-Doo synopsis.

3.2.1 Preprocessing the Input Synopsis

Before information regarding a planning domain model can be extracted from the synopsis, some preprocessing of the natural language input is required. An important aspect of this is to gain a knowledge of the objects that are referenced in the narrative. For this work objects are defined as nouns, provided they are either: proper nouns, concrete nouns, collective nouns, or countable abstract nouns. These objects that appear in the input text will map to objects in the output domain model. Figure 3.2 shows all the objects highlighted for a section of the example synopsis. The first problem to address is how to identify the objects in the natural language input.

It is possible for the same object to be referenced by multiple differing named mentions. In the example both Professor Hyde White and the professor refer to

> The gang travels to the museum (now the next day) where they deliver the knight to the museum curator, Mr. Wickles. He thanks them, but fears that perhaps it wasn't a good idea with Professor Hyde White disappearing. He goes on to explain about the legend of the Black Knight and how it comes to life when the moon is full. Velma asks him what Professor Hyde White was doing with it (despite having already found out beforehand), and replies that the professor was delivering it to the museum all the way from England. As this is going on, they don't notice the knight's glowing eyes. Two workers begin to move the crate, one of them asking Mr. Wickles where to put it. He tells them to put it in the medieval room. As Scooby follows the workers, he finds a strange pair of glasses. He picks them up, as Daphne calls him, while Fred says they're leaving.

Figure 3.2: A section of the example synopsis with the objects highlighted in blue.

the same character. In the full synopsis the character is also referred to using Hyde White and Jameson Hyde White: Prof. of Archaeology. As these all refer to the same character they must be mapped to a single unique identifier which can be used in the output domain model. A stage is therefore needed to disambiguate the objects that have been identified. This requires a method for identifying when multiple named references are referencing the same object and selecting a unique identifier that will be used going forward to represent the object in question. Failing to disambiguate the identified objects would lead to having duplicate objects in the domain model, facilitating the possibility of having objects that don't exist being involved in actions. This wouldn't accurately represent the narrative plot being described by the synopsis.

Given the minimal restrictions placed over the input, it is likely that the synopsis will contain the use of pronominal referencing, i.e., a word used in place of a noun. Many of the pronouns present in the input text will refer to something that has already been mentioned elsewhere in the discourse. Relevant information regarding object properties and the actions that they are involved in, may be conveyed with the use of pronouns. It is therefore important to know which object or objects the pronouns in question are referring to.

Not all pronouns that appear in the input text will be referencing an object. There are many different types of pronoun, some of which may be referencing an object in the story. These pronouns are:

- Personal pronouns - used in place of common and proper nouns, e.g. it.
- Demonstrative pronouns - used to represent a thing or things, e.g. this, that.
- Reflexive pronouns - pronouns that end in -self or -selves, e.g. itself.
- Possessive pronouns - used to show ownership, e.g. its.
- Subject and Object pronouns - used as either the subject or the object in a sentence, e.g. it

It is still possible that these pronouns aren't referencing an object in the story but are instead referring to something else, such as previous actions or events. Some pronoun types will never reference an object in the story and can therefore be ignored. The types of pronoun this applies to are:

- Relative pronouns - used to relate subordinate adjective clauses to the rest of the sentence, e.g. which, that, who, where.
- Indefinite pronouns - used to refer to something unspecified, e.g. all, some, any, several, either.
- Interrogative pronouns - used to ask questions, e.g. who, which, what, where, how.

Deriving the correct interpretation of the text and understanding every reference that is made towards objects in the synopsis is an important problem that has to be addressed in order to extract and create a representative domain model. This is a two part problem: first the pronouns that could potentially be referencing objects in the story have to be identified, and secondly, a knowledge of which objects are being referenced is required going forward. Figure 3.3 highlights the pronouns present in the example that could be referencing objects and need to be addressed.

Coreference Resolution is the task of linking expressions that refer to one another. Coreferencing is an unsolved problem, with no current solution offering 100% accuracy. Existing solutions are available, such as those included with the CoreNLP toolkit [48]. These models represent state of the art solutions and

> The gang travels to the museum (now the next day) where they deliver the knight to the museum curator, Mr. Wickles. He thanks them, but fears that perhaps it wasn't a good idea with Professor Hyde White disappearing. He goes on to explain about the legend of the Black Knight and how it comes to life when the moon is full. Velma asks him what Professor Hyde White was doing with it (despite having already found out beforehand), and replies that the professor was delivering it to the museum all the way from England. As this is going on, they don't notice the knight's glowing eyes. Two workers begin to move the crate, one of them asking Mr. Wickles where to put it. He tells them to put it in the medieval room. As Scooby follows the workers, he finds a strange pair of glasses. He picks them up, as Daphne calls him, while Fred says they're leaving.

Figure 3.3: A section of the example synopsis with pronouns that could be referencing objects highlighted in red. The objects are highlighted in blue.

often achieve accuracy scores in the range of 50-60% [82]. These models however are designed for general-purpose use and therefore the accuracy they can achieve is fairly low. This problem is exacerbated when applied to narrative synopses due to their tendency to use uncommon character and object names that aren't recognised by these general models. Failure to recognise the objects has a detrimental impact on coreferencing, as any reference made towards an unknown object will be missed. This problem is reduced by machine learning approaches that require a training data set. For the purpose of StoryFramer it is possible that no suitable training data already exists for a given synopsis. For the purpose of StoryFramer's specific coreference task, using a method that achieves the highest accuracy and minimises the number of errors is the goal. Based upon the current available coreferencing solutions there is clear motivation to find a method which would be better able to accurately resolve pronouns in the context of a narrative synopsis.

3.2.2 Domain Model Acquisition

The goal is to extract as much planning related information from the synopsis as possible, and from this produce a domain model. This comes with a number of difficulties. The first problem is to define what information described in the input would be of relevance. Subsequently it has to be considered whether the identification and extraction of such information is possible using Information Extraction (IE) techniques in this environment.

With very minimal constraints placed over the input synopsis, the content of each synopsis can vary greatly with regards to the level of detail and information being described. It isn't possible to identify something if it isn't mentioned. The challenge here is to be able to identify when something described in the input is relevant, rather than searching for something specific.

The purpose of a narrative synopsis is to provide a condensed summary, outlining the plot of a play, film or book. Although the level of detail and description can vary, some assumptions can be made regarding the minimum required content that a synopsis should contain. In order to convey the baseline plot, the actions that the story comprises of, and the characters and objects involved in these actions both need to be described. As a minimum, StoryFramer has to be able to search for the verbs in the input sentences that suggest actions performed

> The gang travels to the museum (now the next day) where they deliver the knight to the museum curator, Mr. Wickles. He thanks them, but fears that perhaps it wasn't a good idea with Professor Hyde White disappearing. He goes on to explain about the legend of the Black Knight and how it comes to life when the moon is full. Velma asks him what Professor Hyde White was doing with it (despite having already found out beforehand), and replies that the professor was delivering it to the museum all the way from England. As this is going on, they don't notice the knight's glowing eyes. Two workers begin to move the crate, one of them asking Mr. Wickles where to put it. He tells them to put it in the medieval room. As Scooby follows the workers, he finds a strange pair of glasses. He picks them up, as Daphne calls him, while Fred says they're leaving.

Figure 3.4: A section of the example synopsis, with the actions that occur in the narrative highlighted in red. Verbs that are not describing actions are highlighted in blue.

by characters in the narrative world. Additionally any objects that are participating in said actions should be identified.

Detecting actions in natural language is a difficult task, with their interpretation often depending on contextual information. To illustrate this, Figure 3.4 highlights the verbs that could be conveying actions in the example text. Verbs that are describing actions that take place in the narrative are highlighted in red. Highlighted in blue are the verbs that could potentially be describing actions but based on their context are not. In this example, these verbs instead are used to: describe emotions; recall past events; narrate the story; describe properties; ask questions; give instructions; and are used in other forms of speech. Here the role of StoryFramer is to identify plausible candidate verbs that will be mapped to domain model operators. All of the highlighted verbs could become actions in the domain model. Whether it is possible to distinguish between those that are judged to have actually occurred in the narrative, and those that have not based on their context is a problem to be addressed. Alternatively, limiting the domain model to only the actions that actually occurred may not be desirable as this would also limit the possible story variants that such a model could produce.

Framer [44] showed that to in order to acquire planning domain models from natural language action descriptions, the input had to adhere to some strict con-

ditions on how the sentences were written. Given that the input for StoryFramer are narrative synopses that could be sourced from the internet, forcing strict constraints over the input isn't a plausible option. Extracting all the information available in the synopsis that fits within StoryFramer's selection criteria allows for a domain model to be partially built. This can then be completed by an author with domain modelling expertise. This presents a trade-off, the input required of an author could be minimised at the cost of domain model flexibility.

3.3 StoryFramer Overview

The task at hand presents many problems and challenges, and following their analysis a suitable approach referred to as StoryFramer has been developed. StoryFramer is a semi-automated approach that consists of six major components that act as stages of a pipeline, with the output of one component required as input for the next. The components can be further categorised into two main stages: "Preprocessing Input Synopsis" and "Domain Model Acquisition". Each component contains a number of smaller components. Together these compo-

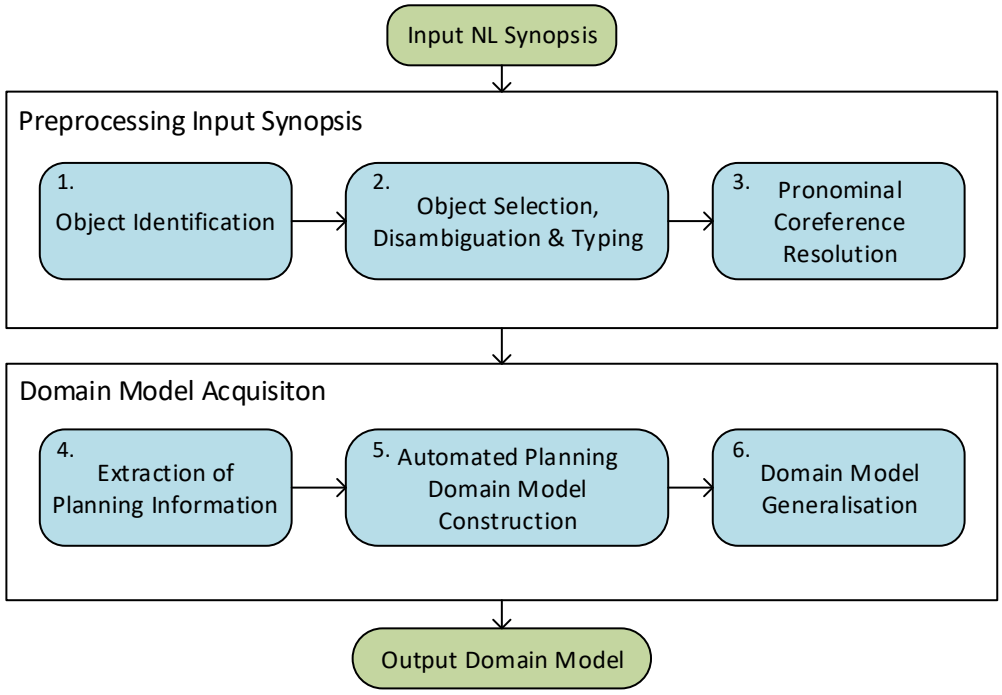

Figure 3.5: An overview of the StoryFramer approach showing its constituent components and how they are ordered.

nents facilitate StoryFramer's overall goal: to take an input natural language plot synopsis and from this, construct a planning domain model representative of the input. An overview of the StoryFramer pipeline is shown in Figure 3.5. The major components are described in detail in the following chapters. The "Preprocessing Input Synopsis" section of the figure is the focus of Chapter 4 and the "Domain Model Acquisition" section is the focus of Chapter 5. A brief overview of each component is presented in Section 3.3.1. The role of the author is discussed in Section 3.3.2.

3.3.1 An Overview of the StoryFramer Components

1) Object Identification - The automated identification of named object mentions (references referring to characters, objects and things) that are present in the input synopsis.

2) Object Selection, Disambiguation & Typing - Objects are selected from the identified object mentions and are disambiguated to create a list of unique objects. Suitable types are then assigned to these objects. These tasks are completed by the author.

3) Pronominal Coreference Resolution - The automated coreference resolution of pronouns that have been used to reference an object(s).

4) Extraction of Planning Information - The automated identification of the narrative events that are described by the synopsis. This included the identification of objects that are participating or are associated with each event and the identification of additional properties.

5) Automated Planning Domain Model Construction - The automated construction of a default planning model that is representative of the extracted planning information. A planning model capable of reproducing the original plot can also be automatically constructed.

6) Domain Model Generalisation - The default planning model can be generalised by the author. Amendments can be made to the model using the default predicates and control that the default model provides. A planning model capable of generating new story variants can be created by making such amendments.

3.3.2 The Role of the Author

The author that is using the StoryFramer approach is utilised throughout in order to create a planning model that accurately represents an input synopsis. The role of the author can be defined as the fulfilment of three tasks: validating automated processes; providing additional information; and making preferential choices. These tasks can be completed by non-technical authors as no expertise of modelling planning domains is required.

In order to create a planning model representative of an input synopsis it is important that the results of each task are correct. Accuracy can be ensured by having the author validate the results of StoryFramer's automated processes. Any errors that are incurred as a result of an automated process can be corrected by the author, which also prevents errors from propagating down the pipeline.

Additional information such as the types of object that are present in the synopsis can be provided by the author. This information can then be exploited by the automated processes to achieve a significant increase in accuracy. Such information is either already known to the author or can be acquired through reading of the input synopsis.

A number of preferential choices are presented to the author during the StoryFramer approach. For some tasks there are multiple correct ways in which they can be completed. One such task is the selection and disambiguation of objects. The author groups object mentions together based upon the unique objects that they are referencing. What is considered to be an object is a decision that can be made by the author, which in turn has an effect on the completion of this task. Another task where author preference has to be considered is the generalisation of the default planning model. The amendments that are made to the model in order to produce a model capable of generating new story variants are dependant on the types of new story that the author wishes to generate. This task is therefore completed by the author.

Chapter 4

Preprocessing Input Synopses

This chapter details the first stage of the StoryFramer approach to domain model acquisition from input synopses; the preprocessing of input synopses. Common features of narrative synopses are analysed and discussed. The available natural language processing tools and annotations are introduced, detailing how they can be used in this context. Preprocessing input synopses comprises of three main tasks: the identification of objects within the input text; object selection, disambiguation and typing; and pronominal coreference resolution. Methods are presented that provide solutions to these tasks, exploiting the NLP tools and the information available. The approach taken will be discussed and demonstrated using the Scooby-Doo example synopsis introduced in Section 3.1.

4.1 Feature Analysis of Narrative Synopses

One important goal of StoryFramer is to place as few restrictions on the input synopses as possible. One of the research questions being answered is that of; can you take any natural language narrative synopsis and use that as a basis for domain model acquisition? In order to address this, the general features of narrative synopses need to be considered, as well as the minimum set of requirements/restrictions that the input has to adhere to.

4.1.1 Input Requirements and Assumptions

For a narrative synopsis to be used as input in this work it has to meet the following to two requirements: The synopsis has to be written in third-person; and written using the English language.

Narrative synopses should be written in third-person and present tense, but when using sources that have been written by a human there can be no guarantee of this. One of the assumptions that can be made of third-person synopses is that the first time any object/person/thing is referenced, it will be referenced by name. This doesn't always hold true of first-person written pieces, where there is no requirement for the narrators identity to be stated in the natural language, often their identity stems from source context, e.g., the author. Knowing the identities of the 'things' that are present in the synopsis is important for constructing a domain model, as it allows for the tracking of an object's involvement in actions. It would be possible to relax the third-person requirement with the trade-off that an author will assume the burden of ensuring that all 'things' referenced in the synopsis have the correct identities.

The process of extracting planning domain models from synopses isn't restricted to the English language. In this work English is used mainly as a preference but also due to its compatibility with the NLP tools that are being utilised. In order to adapt the method to another language, the NLP annotations being used would need to be trained on models for the appropriate language. Additionally the rules used for coreferencing pronouns may have to be modified to achieve more accurate results. A fair assumption being made is that the NL sentences of the synopsis are well-formed and in accordance with a language's grammar. In order to extract the narrative information required the synopsis has to at least contain such information regarding the narrative events that occur and be presented in a clear, understandable manner. If the input fails to achieve this, the synopsis isn't a suitable source for domain model acquisition.

4.1.2 Common Features of Narrative Synopses

The content of synopses can greatly vary, but there are a number of common features that they will all likely contain.

The purpose of a narrative synopsis is to provide a plot summary of a story. In order to do this the minimum requirement of the content is to include information regarding the characters and objects involved, and how they participate in the narrative actions that occur. The level of detail that they are both described with vary, but the core purpose will be the same. A well-written synopsis is not just a dry list of events and should include actions, revelations and emotions [50].

Although not a requirement, it is likely that a synopsis will describe how characters are feeling and how they are motivated to carry out specific actions.

Another common feature of narrative synopses is the use of pronominal referencing, i.e, the usage of pronouns. Pronouns are commonly used in writing to avoid the repetition of nouns. Pronouns are used to reference someone or something that has previously been mentioned in the discourse. The assumption is made that all pronouns have been used correctly so that the antecedent(s) being referenced are clear and there is no ambiguity.

Similar to how pronouns are used to avoid the repetition of nouns, an alternate and common approach is to use multiple named references for the same person or object. When this is the case it is expected that the multiple named mentions are used clearly in such a way that it is easy to determine that they are referring to the same thing. A simple example would be when a character's first name is used, as opposed to their full name and title.

4.1.3 Genres and Sources

In order to demonstrate that the method presented as StoryFramer is applicable to the wide range of narrative synopses that are being targeted, a selection of synopses that represent a variety of genres and have been taken from a number of differing sources are used for evaluating this work. The synopses used are shown in Fig 4.1. The content of each synopsis is dependent on both the author and its target audience. Differing genres may also utilise unique styles and vocabulary that introduce a new set of problems.

Synopsis	Format	Synopsis Source
Scooby-Doo (1969)	TV episode	http://scoobydoo.wikia.com
Friends (2003)	TV episode	https://www.imdb.com
House (2004)	TV episode	http://house.wikia.com
The Jungle Book (1967)	Film	https://en.wikipedia.org
Toy Story (1996)	Film	https://en.wikipedia.org
Titanic (1997)	Film	https://en.wikipedia.org
Merchant of Venice (~1605)	Play	https://www.nosweatshakespeare.com
A Christmas Carol (1843)	Novel	https://www.sparknotes.com
Lord of the Flies (1954)	Novel	https://www.sparknotes.com
Odyssey (~675-725 BC)	Poem	https://www.sparknotes.com

Figure 4.1: A table of the synopses used in this work.

4.2 Natural Language Processing Tools and Annotations

For the task of preprocessing synopses, a number of Natural Language Processing (NLP) techniques are used. NLP toolkits, such as Stanford CoreNLP [48] provide an array of natural language annotations and analysis tools.

The methods in this chapter utilise three types of annotation: Part-of-Speech (POS) tagging; syntactic constituency analysis; and typed dependency parsing. These annotations will be detailed and illustrated using the example sentence below.

As Scooby follows the workers, he finds a strange pair of glasses.

Before any analysis can take place, the input sentence has to first be tokenized. This is the process of converting a sequence of characters into a sequence of tokens (strings with an identified meaning). In this case, taking an input sentence and converting it into a sequence of words and punctuation as follows:

[As] [Scooby] [follows] [the] [workers] [,] [he] [finds] [a] [strange] [pair] [of] [glasses] [.]

4.2.1 Part-of-Speech Tagging

Part-of-Speech (POS) tagging is the process of marking up tokens corresponding to particular parts of speech, based upon definition and context, i.e., relationships with adjacent and related words. Tokens are labelled with POS tags which indicate the token's part of speech and often other grammatical categories such as (case, tense, etc.)

The Penn Treebank [49] is an annotated corpus of English that present a simplified POS tagset that is widely used in NLP, including the CoreNLP toolkit. The Penn Treebank tagset is used in this work and the more POS tags of interest are: VB (verbs), NN (nouns) and JJ (adjectives). These tags can also be extended to include more information, for example, NN (noun, singular or mass) can be extended to become: NNS (noun, plural), NNP (proper noun, singular) and NNPS (proper noun, plural).

POS tagging was once a task performed by hand but is now done in the context of computational linguistics. Using the POS tagger provided with the

CoreNLP toolkit on the example sentence yields the result below. In addition to nouns (/NN), verbs (/VB) and adjectives(/JJ); this sentence also includes propositions (/IN), determiners (/DT) and pronouns (/PRP).

[As/IN] [Scooby/NNP] [follows/VBZ] [the/DT] [workers/NNS] [,/,] [he/PRP] [finds/VBZ] [a/DT] [strange/JJ] [pair/NN] [of/IN] [glasses/NNS] [./.]

4.2.2 Syntactic Constituency Analysis

Constituency parsing breaks a text down into sub-phrases. In the syntactic analysis of linguistics, a phrase is a word or a group of words that functions as a single unit within a grammatical hierarchy. Most phrases contain a key word that identifies the type and linguistic features of the phrases; also know as the head-word. The syntactic category of the head-word is used as the name for the category of the phrase; for example, a phrase whose head-word is a noun is called a noun phrase.

Figure 4.2 shows a visual representation of a constituency parse tree for the example sentence. This is a context-free phrase representation of the text. The phrasal nodes are highlights in green, with POS tags in blue, and the tokens of the sentence in white.

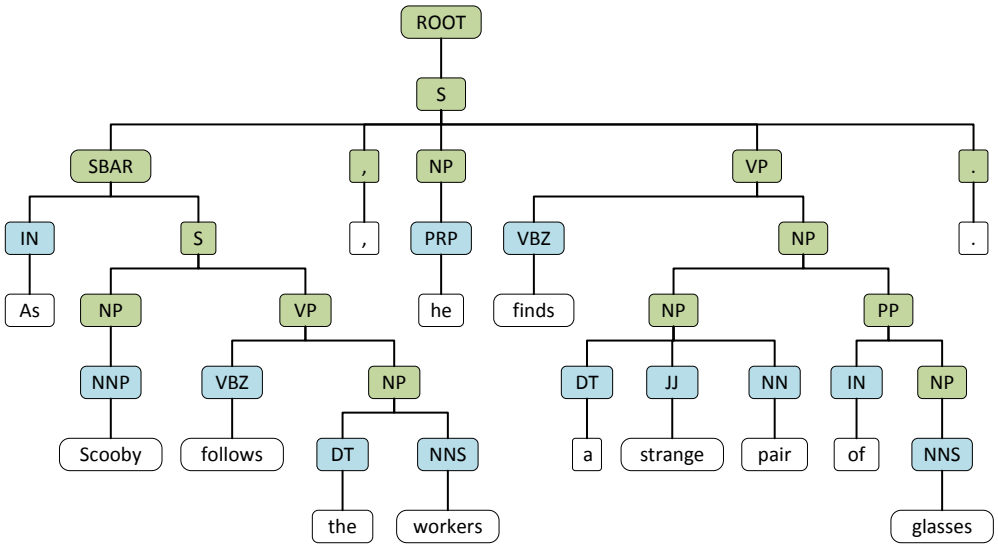

Figure 4.2: A Constituency Parse Tree for the example sentence.

4.2.3 Typed Dependency Parsing

Dependency parsing analyses the grammatical structure of a sentence, establishing relationships between head-words and words which modify those heads. The (finite) verb is taken to be the structural centre of clause structure, with all other words either directly or indirectly connected to the verb by links, called dependencies. Dependency grammar representations are distinct from constituency grammars, as they lack phrasal nodes. The structure is determined by the relation between a word and its head.

CoreNLP by default uses the Stanford typed dependencies representation [20] to provide a simple description of the grammatical relationships in a sentence. Figure 4.3 presents a dependency parse tree and diagram that uses the Stanford typed dependency representation. The dependency between a dependent and its head-word (parent on the tree) is highlighted in red. Dependencies are used to label grammatical relationships such as the subjects and objects of clauses, modifiers and conjunctions, amongst others. In the example, *Scooby* is the nominal subject (nsubj) of the clause *follows*.

Dependency Parse Tree

```
-> finds / VBZ  (root)
    -> follows / VBZ  (advcl)
        -> As / IN  (mark)
        -> Scooby / NNP  (nsubj)
        -> workers / NNS  (dobj)
            -> the / DT  (det)
    -> , / ,  (punct)
    -> he / PRP  (nsubj)
    -> pair / NN  (dobj)
        -> a / DT  (det)
        -> strange / JJ  (amod)
        -> glasses / NNS  (nmod:of)
            -> of / IN  (case)
    -> . / .  (punct)
```

Dependency Diagram

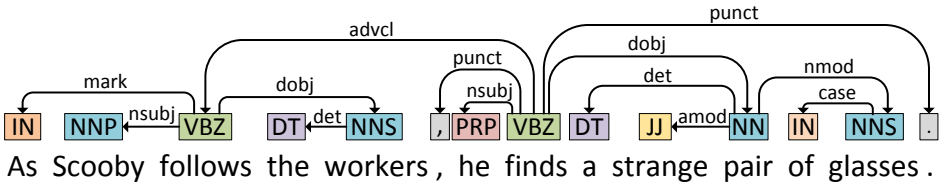

Figure 4.3: A Typed Dependency Parse Tree and Diagram for the example sentence.

4.3 Identifying Objects Within Input Synopses

As discussed in Section 4.1, the level of detail that a synopsis describes a narrative in can vary greatly. However, with the core purpose of a synopsis being to convey a plot summary, as a minimum the audience has to be made aware of the characters and objects that appear in the story, and how they are involved and participate in the narrative events that occur. The identification of this information is the basis for extracting a planning domain model.

4.3.1 Definition of an Object

In order to identify the objects mentioned in the input text, what is considered an object in this work needs to be defined. In this work all nouns are considered objects. The following list provides definitions for the different noun types (definitions taken from [62]):

- Proper nouns - a name identifying a particular person, place, organisation or thing, e.g. Fred, London, Google, Monday.
- Concrete nouns - referring to people and things that exist physically and can be seen, touched, heard, tasted or smelled, e.g. dog, coffee, rain, tune.
- Collective nouns - referring to groups of people or things, e.g. gang, team, government, herd.
- Abstract nouns - abstract nouns refer to ideas, qualities and conditions which have no physical reality. They can be further categorised into those of which that are countable (often has a plural form), e.g. ideas, skills, problems, mistakes; and uncountable abstract nouns, e.g. happiness, danger, love, news.

The objects identified suggest likely objects that could be used to populate output planning domain models, but are also used during the coreference resolution of pronouns. Because it is possible to use a pronoun to reference any noun, all nouns should be considered objects such that the correct references can be identified. Objects that are mapped to the domain model will be able to participate in actions if they meet the parameter requirements. There is no clear right

and wrong answer when it comes to which nouns should be represented as objects in the domain model, this is decided by the domain author. A case could be made that any noun could be used as an object in a domain model. Rather than limit the options available in an extracted domain model, the approach taken is to identify as many objects as possible.

4.3.2 Identifying Objects in Natural Language

To identify objects in the input synopsis an approach that utilises Natural Language Processing (NLP) techniques is required. In this section the method that StoryFramer uses for this task is presented. The method makes use of the NLP annotations described in section 4.2.

Object Identification Method

Given an input narrative synopsis, objects are identified on a sentence by sentence basis. Using the CoreNLP toolkit [48], POS and syntactic constituency parse tree annotations are produced. The POS annotation labels each token of the sentence with a part-of-speech tag. The syntactic constituency parse tree

Algorithm 1: Object Identification Algorithm

```
Function Main(Input):
    // Input is a constituency parse tree
    for n in Nodes(Input) do
        if n is NP then
            CheckChildren(n)
        end
    end

Function CheckChildren(node):
    children = Children(node)
    for child in children do
        if child is NOUN then
            MakeCandidateObject(child)
        end
        if child is ADJECTIVE and NoNouns(children) then
            // None of the children are nouns
            MakeCandidateObject(child)
        end
    end
```

provides a context-free phrasal representation of the sentence structure. The labelled POS tags are available in the constituency parse tree, the parent node of each token node (excluding punctuation) is a POS node.

Starting with the root node of the constituency parse tree, a recursive search is used to check every node in the tree from left to right. Every time a noun phrase (NP) node is found, the children are checked to see if any are nouns or adjectives. All children that are nouns become candidate objects. If an adjective is found and none of the other children are nouns, the adjective becomes a candidate object. This method is shown in Algorithm 1.

Object Identification Example

Figure 4.4 shows a syntactic constituency parse tree for the following sentence: *A man is driving a pick-up down a road.*

Using the method presented in Algorithm 1, the parse tree is searched, looking for every noun phrasal (NP) node. On finding a noun phrase node, the children of the noun phrase node are then checked for potential objects. Any nouns that are found become candidate objects: (*man / NN*) and (*road / NN*). If a noun phrase node's children doesn't contain a noun, adjectives are then searched for and become candidate objects if found: (*pick-up / JJ*).

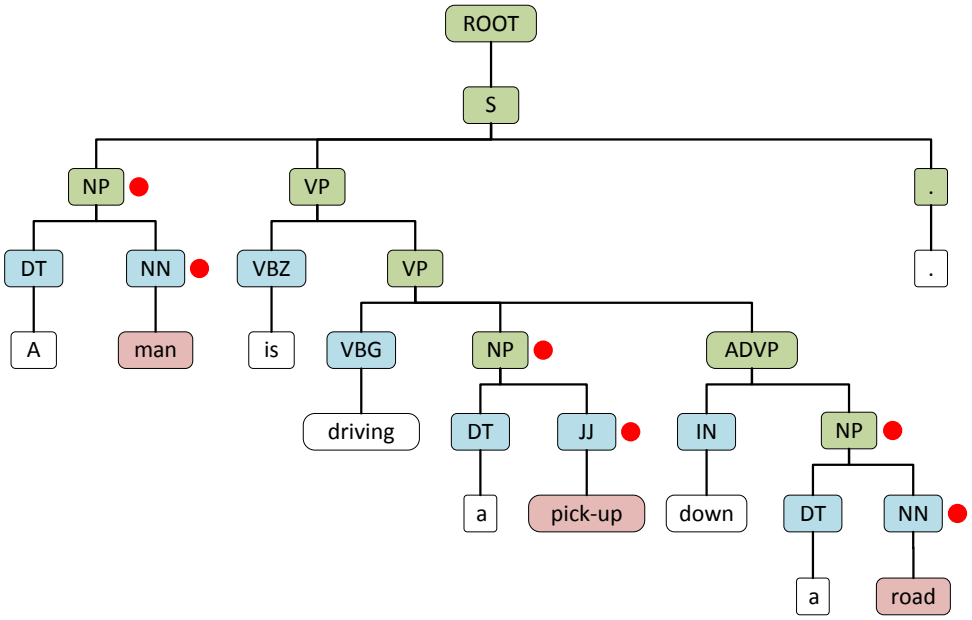

Figure 4.4: A Constituency Parse Tree for the sentence, *A man is driving a pick-up down a road.* The red dots indicate the nodes that the object identification method searches for.

4.3.3 Extracting Object Names

The object identification method finds tokens (words) that indicate an object mention. There is no restriction however that object names are limited to singular words. Such a restriction would reduce the level of detail of which object names are identified. It is often possible that a more accurate detailed object name can be extracted from the sentence by looking to the other surrounding words. This is why words are first identified only as candidate objects. For example, an adjective might be used as a way of identifying a specific object, e.g. “Red Truck” and “Blue Truck”. The words that surround the candidate word can provide more information about the object and affect the word’s meaning.

It is also possible that two candidate object words both refer to the same object. If this is the case only one object should be identified. Using Figure 4.5 as an example, both the words ’pair’ and ’glasses’ are identified as candidate objects, however both words contribute to the name of one object, ’a pair of glasses’. Additionally the adjective ’strange’ has been used to differentiate these glasses from a regular pair and should also be included in the extracted objects name. The ideal and most detailed name that could be extracted in this case

would be, *strange pair of glasses.*

Extracting Object Names Method

In order to extract detailed object names, the grammatical relationships between words (available through typed dependency parsing) are used. As a result of the previous object identification method, some tokens may have been flagged as candidate objects. Every node in a dependency parse tree represents a token in the NL sentence. This method locates any flagged candidate object tokens on the dependency parse tree and checks the grammatical relationships of the candidate object node to determine whether a more detailed object name can be extracted. An outline algorithm for extracting object names is shown in Algorithm 2.

Starting with the dependency parse tree annotation for a given sentence, the nodes in the tree are checked using a recursive search. When a node that has been identified as a candidate object is reached, the relationships between the

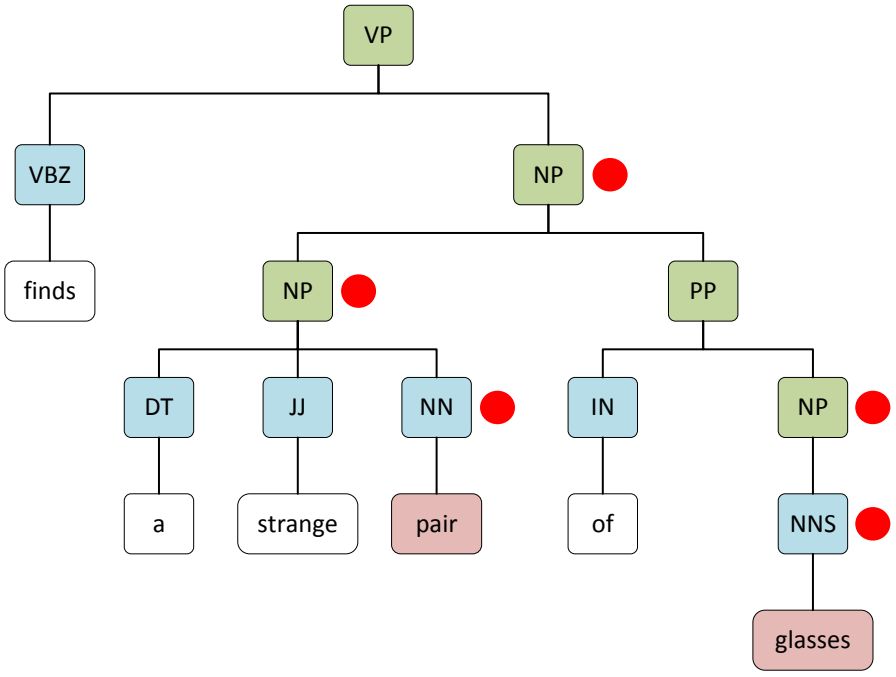

Figure 4.5: A section of the syntactic parse tree for the example sentence, *As Scooby follows the workers, he finds a strange pair of glasses*. The full parse tree is shown in Figure 4.2. The red dots indicate the nodes that the object identification method searches for, resulting in both *pair* and *glasses* being identified as candidate objects.

Algorithm 2: Object Naming Algorithm

```
Function Main(Input):
    // Input is a dependency parse tree
    CheckChildren (Input.RootNode)
Function CheckChildren(node):
    children = Children(node)
    for child in children do
        if child is CANDIDATE OBJECT then
            CheckNameModifiers (child)
        end
        if Children (child) > 0 then
            // Recursively check every node
            CheckChildren (child)
        end
    end
```

node and its children are checked to see if any of the following dependencies are found:

- Noun Compound (compound) - the compound word is added to the object name in the order that they appear in the text. This applies for multiple compound children.
- Adjectival Modifier (amod) - the adjective modifying the object is added in front of the object name.
- Nominal Modifying Of (nmod:of) - “of” and the child word is added onto the end of the object name. If a determiner exists between the of and the child word, it too is added.

It is possible that one of the children that modify a candidate object’s name is also a candidate object. In this situation, before the child is included in the name of the object, the same dependency naming checks are applied to its children. The resulting object name is then added to the original candidate’s name (its parent) in the appropriate manner. Because the child has been included in another object’s name, it is no longer recognised as a candidate object is now unflagged; extracting duplicate objects is therefore avoided.

Extracting Object Names Example

Figure 4.6 shows a dependency parse tree for the example sentence, *As Scooby follows the workers, he finds a strange pair of glasses*.

Starting with the the root node (finds), the tree is searched for candidate objects. The first candidate objects encountered are Scooby and workers. Dependency name checks are applied to the children of both but no name-altering dependencies are found. The objects are extracted as *SCOOBY* and *WORKERS*.

The next candidate object to be reached and checked is pair. Checking the node's children, two name-altering dependencies are found: strange/JJ (amod); and glasses/NNS (nmod:of). On finding an amod, the word is added to the front of the object name: (strange + pair). For the nmod:of, the word is added to the end of the object whilst retaining the "of": (pair + of + glasses). The object is extracted as *STRANGE PAIR OF GLASSES*.

At the start of the object naming algorithm, glasses was flagged as a candidate object. Before the recursive search reached the glasses node it had contributed towards the naming of pair, and in doing so the candidate object flag was removed. When the node is reached, no duplicate object is extracted.

Figure 4.7 contains two more examples of the object naming method in prac-

```
-> finds/VBZ (root)                    Extracted Objects
    -> follows/VBZ (advcl)
        -> As/IN (mark)
        -> Scooby/NNP (nsubj)          - Scooby
        -> workers/NNS (dobj)          - workers
            -> the/DT (det)
    -> ,/, (punct)
    -> he/PRP (nsubj)
    -> pair/NN (dobj)                  - strange pair
        -> a/DT (det)                    of glasses
        -> strange/JJ (amod)
        -> glasses/NNS (nmod:of)
            -> of/IN (case)
    -> ./. (punct)
```

Figure 4.6: A Dependency Parse Tree for the example sentence. The candidate objects are highlighted in red. Words and dependency relations involved in the extraction of object names are highlighted in blue.

tice. The first of which illustrates how noun compound dependency relationships are handled using *Professor Hyde White* as the example. As previously explained, the order in which the two compound words (Professor and Hyde) are affixed to the candidate object parent (White) is based upon the order in which they appear in the NL sentence. The second example shows that not all noun modifiers contribute to an object's name. In this case the (nmod:poss) dependency indicates that the object is in the possession of another object, thus two different objects (suit of armour and knight) are extracted.

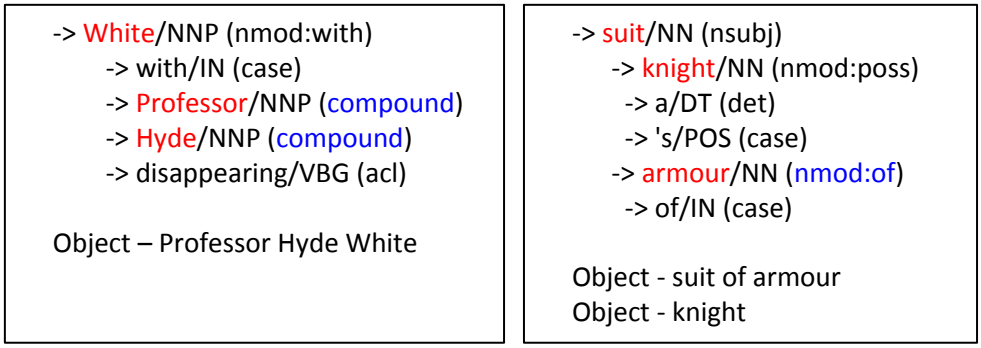

Figure 4.7: Object name extraction examples

4.4 Object Selection, Disambiguation and Typing

After all the object mentions that appear in the input sentences have been identified, the next processing step concerns the selection, disambiguation and typing of these object mentions. The result of doing so is a typed list of the unique objects that are present in the synopsis.

4.4.1 Object Selection

StoryFramer is an aid to creating planning domain models, for which there is no one definitive correct approach. An aspect of creating planning domain models is author preference. How the author intends for the domain model to operate will be a contributing factor when deciding which objects need to be represented in the model; and consequently how the mentions should be clustered. For this reason the task of object selection and consequently the clustering of object mentions is handled by the author. While author preference has a factor in the selection of objects, in order to ensure the output domain model is representative of the input synopsis objects should be included if any of the following are true: 1) The object represents a character or group of characters that are involved in an event; 2) The object is referenced by a pronoun; or 3) The coreference of a pronoun is dependent on the object being recognised as such (detailed in Section 4.5.5).

4.4.2 Object Disambiguation

Each unique object mention identified in the text doesn't necessarily correspond to a unique object. It is possible that an object is referenced in the text using multiple differing named mentions. In the example synopsis, 'Professor Hyde White' is also referred to using: 'the professor', 'Hyde White' and 'Jameson Hyde White: Prof. of Archaeology'. With all of these extracted object mentions referring to the same object, it would be incorrect to have these mentions mapped to more than one object in the output domain model. So that each individual object present in the synopsis is mapped to a singular object in the domain model, object mentions that refer to the same object need to be clustered and given a unique identifier.

Identified Object Mentions		Example Object Clustering	
Shaggy	Mr. Wickles	- Shaggy	- Museum
Scooby	Museum Curator	- Scooby	County Museum
Suit of armour	Professor Hyde White	- Black Knight	- Gang
Knight	Black Knight	Knight	- Mr. Wickles
Fred	Professor	Suit of armour	Museum Curator
Velma	Workers	- Fred	- Workers
Daphne	Crate	- Velma	- Crate
London	Medieval room	- Daphne	- Medieval room
Prof. of Archaeology	Strange pair of glasses	- London	- Strange pair of glasses
Jameson Hyde White	Glasses	- Professor Hyde White	Glasses
Hyde White	Library	Prof. of Archaeology	- Library
County Museum		Jameson Hyde White	
Gang		Hyde White	
Museum		Professor	

Figure 4.8: An example object clustering. Sorting a list of identified object mentions into a disambiguated list of unique object clusters.

The Disambiguation Process

Starting with the list of identified object mentions, the author will create an object cluster to represent each unique object that appears in the the input synopsis. Object clusters comprise of all the named mentions that appear in the text that are used to reference the same object. For clusters that contain multiple object mentions, one of the mentions is selected to act as the object's unique identifier going forward. The table shown in Figure 4.8 provides an example object clustering that disambiguates object mentions identified in the Scooby-Doo example synopsis.

Collective nouns should be viewed as their own objects for the purposes of this clustering process. It is however important to understand which individual objects are being referenced when a collective noun is used. A reference to a collective noun, is a reference to all of the objects it references. The opposite however isn't true; a reference to an object that is part of a collective, isn't a reference to the collective itself. In the Scooy-Doo example, the five main characters are often referred to as a collective group, i.e. "The Gang." `Gang` becomes an object reference that when used refers to the five members. A reference to the `Gang` is a reference to `Scooby-Doo` as he is one of the members, but a reference to `Scooby-Doo` doesn't reference the collective `Gang`.

4.4.3 Object Typing

Through the clustering of object mentions, the objects that appear in the input and the named object mentions that are used to reference them have been established. Synopses may include cases of pronominal referencing, i.e. the use of pronouns. Determining the object or objects that are being referenced when a pronoun has been used requires a knowledge of an object's type.

Object typing is also used in modelling planning domains as it allows for type constraints to be applied to action parameters, providing a mechanic for controlling the eligibility of object participants.

Five types are used in this work that provide sufficient information such that the coreferencing of pronouns and some control over action eligibility in the planning domain can be achieved. The specifics of how these types are used will be detailed in the later relevant sections. The types are as follows:

- MCHAR - Male Character
- FCHAR - Female Character
- OTHER - Object/Location/Neutral (Singular)
- OTHERP - Object/Location/Neutral (Plural)
- GROUP - Group or Organisation

The Object Typing Process

Author preference once again plays a part in the typing of the objects, as how the object mentions were clustered may be contributing factor when assigning a type. The author will assign one or more of the listed types to each object. The types given to an object should reflect the role that the object plays in the synopsis and should also be compatible with the pronouns that are used to reference the object in the text. When assigning multiple types to an object, the object gains the eligibility that each type provides. By assigning multiple types, the object can also be referenced by pronouns that are grammatically compatible with each of the types selected. Figure 4.9 shows an example of how the objects in Scooby-Doo could be typed.

Object	Type		Object (cont.)	Type
- Shaggy	MCHAR		- Museum	OTHER
- Scooby	MCHAR		County Museum	
- Black Knight	MCHAR/OTHER		- Gang	GROUP
Knight			- Mr. Wickles	MCHAR
Suit of armour			Museum Curator	
- Fred	MCHAR		- Workers	GROUP
- Velma	FCHAR		- Crate	OTHER
- Daphne	FCHAR		- Medieval room	OTHER
- London	OTHER		- Strange pair of glasses	OTHERP
- Professor Hyde White	MCHAR		Glasses	
Prof. of Archaeology			- Library	OTHER
Jameson Hyde White				
Hyde White				
Professor				

Figure 4.9: An example of object typing using the Scooby-Doo object clusters from Figure 4.8.

In the example given, the task of typing the objects in most cases is trivial for an author providing they have read the synopsis. The *BlackKnight* object is a case where it is appropriate to assign multiple types to an object. In this example the object was clustered such that the object represents both states that the suit of armour can be in (alive and lifeless). As this is the case the object is typed to cater for both these states, becoming both a male character (MCHAR) and an object (OTHER). The sentence below shows that the object is also referenced using both male (his) and neutral (its) pronouns, providing additional confirmation that this typing is warranted.

The suit of armour in the back has come to life and left his containment.
Its eyes glow yellow from inside its helmet.

The author alternatively could have chosen to represent the two states of the *BlackKnight* and *SuitOfArmour* as separate objects. In this case a suitable typing would be: *BlackKnight* (MCHAR); and *SuitOfArmour* (MCHAR/OTHER). *SuitOfArmour* is still referenced by male pronouns so would still require the compatible male typing. With the typing of objects dependent on the author's clustering preferences, the responsibility to assign types that are consistent with the object clusters lies with the author.

4.5 Pronominal Coreference Resolution

Pronouns, i.e. words used in place of nouns, are widely used in natural language to avoid repeating nouns. Pronouns refer to someone or something that has been mentioned elsewhere in the discourse, this is also known as pronominal referencing.

With minimal restrictions placed over the synopses that StoryFramer can take as input, it is very likely that these will include occurrences of pronominal referencing. In order to correctly extract information from sentences that utilise pronominal referencing, an understanding of the objects or things that are being referenced is required.

4.5.1 Exploiting the Available Information

Coreference Resolution is the task of finding all expressions that refer to the same entity in the text. Pronouns and other referring expressions are linked to entities. Coreference resolution is a complex problem that has yet to be solved, meaning no perfect solution exists. General solutions such as those provided by the CoreNLP toolkit offer an accuracy of 50-60% [82].

The way these algorithms commonly function is to find the first preceding entity that is compatible with the expression. If a preceding entity hasn't been detected or any information regarding it's type is unknown, this will often result in the correct coreference link not being found. In the context of narrative synopses this can be a more prevalent issue, with characters and objects often being referred to using unique names that CoreNLP will regularly fail to recognise, e.g., the character *House* in House M.D. is recognised as an organisation rather than a male character.

In the context of StoryFramer, information regarding the objects that are present in the input synopsis and their types is available in the form of the typed object list. By developing a pronominal coreferencing algorithm that exploits this available information, errors that are caused by a failure to recognise an object can be avoided. The goal of StoryFramer's pronominal coreferencing approach is to achieve fewer errors than using a general CoreNLP solution in the given context.

4.5.2 Pronoun Types That Require Coreferencing

The assumption is made that the input narrative synopses will be written in third-person and present tense (as discussed in Section 4.1). As this is the case, the types of pronoun that can appear in the text becomes restricted to: pronouns that can potentially be referencing objects in the narrative; and those that cannot. The pronouns that can potentially be referencing objects and require coreferencing are as follows:

- Subject/Subjective Pronouns - Pronouns that act as the subject of a clause or sentence. (He, She, It, They)
- Object/Objective Pronouns - Pronouns that act as the object of a clause or sentence. (Him, Her, It, Them)
- Possessive Adjectives - Possessive adjectives show ownership. These pronouns are technically adjectives because they modify a noun that follows them. (His, Her, Its, Their)
- Possessive Pronouns - Pronouns that also show ownership, however these pronouns refer to a previously named or understood noun. (His, Hers, Its, Theirs)
- Reflexive Pronouns - Object pronouns that are used when the subject and object are the same noun. (Himself, Herself, Itself, Themselves)

For a given pronoun, its type often has an effect on the coreferencing process. In most cases, depending on the pronoun type and what noun is being referenced a unique word is used. This makes identifying most types a simple task. There are however cases where the same word has been used for multiple pronoun types. One of the most common examples of this is the female reference “Her” which can be used for both a possessive adjective and an objective pronoun. In contrast male references are more straightforward with “His” used for a possessive adjective and “Him” for an objective pronoun. This creates the need for a method that can differentiate between the two types. Possessive adjectives are named as such because they modify a noun. By looking at the part-of-speech tags for the next word(s) following the pronoun it is possible to interpret whether the pronoun is possessive or not. A “Her” followed by a noun or a number of adjectives that are also modifying a noun is a possessive adjective.

4.5.3 Pronoun and Object Type Compatibility

Antecedents are words or phrases to which another word (often a relative pronoun) refers back to. Coreference algorithms commonly work by finding the nearest antecedents to a pronoun and checking to see if any are a compatible match with the pronoun. In order to do this it has to be known which pronouns and antecedents are compatible with one another. In the context of StoryFramer, the antecedents are the objects in the synopsis that have already been identified, typed and stored in a typed object list (Section 4.4.3). Every object will have been assigned at least one type. The compatibility between pronouns and object types is:

- MCHAR: He, His, Him, Himself.
- FCHAR: She, Her, Hers, Herself.
- OTHER: It, Its, Itself.
- OTHERP: They, Their, Them, Themselves.
- GROUP: They, Their, Them, Themselves.

In cases where an object has multiple types, if one of the types is compatible with the pronoun being resolved, the antecedent is deemed compatible.

4.5.4 Identifying Sentence Clauses

As part of the coreferencing process the input sentences need to be broken down into their constituent clauses. In this work a sentence break is defined as a point separating two clauses within the same sentence. Sentence breaks are identified by searching for the following:

- Punctuation - commas, semicolons and colons.
- Coordinating conjunctions - words such as “but” and “and.”

Figure 4.10 shows a section of the example synopsis, broken down into numbered clauses and highlighting where sentence breaks have been identified. There are three exceptions that result in a break no being identified despite meeting the punctuation or coordinating conjunction criteria. The first is when a

```
[Fred comments on why a knight's suit of armour would be out
alone in the middle of the night.1][Shaggy jokes that maybe he's
out for the night.2][Velma chides him for his joke3]|,|[as Daphne
wonders who it belongs to.4][Fred reads5]|:|["Deliver to Jameson
Hyde White: Prof. of Archaeology6]|,|[London7]|,|[England."8]
[Shaggy makes another joke about having heard of hide and
seek9]|,|[but not "Hyde White".10][Velma says that is an English
name.11][Daphne also finds a delivery slip reading12]|:|["Deliver
to the County Museum."13][The gang travels to the museum (now the
next day) where they deliver the knight to the museum
curator14]|,|[Mr. Wickles.15]
[He thanks them16]|,|[but fears that perhaps it wasn't a good
idea with Professor Hyde White disappearing.17]
```

Figure 4.10: A section of the Scooby-Doo example synopsis broken down into numbered clauses. Identified sentence breaks are highlighted between two red lines, | break |.

coordinating conjunction is being used to join two words together, e.g. “hide and seek,” (Clause 9). A coordinating conjunction is also not considered to be a sentence break if it directly follows punctuation that is already considered to be a break (“but” in Clause 17). If an object name contains punctuation or a coordinating conjunction, these are also not identified as sentence breaks (“Jameson Hyde White: Prof. of Archaeology” in Clause 6).

Clauses aren’t accurately identified when using this method from a grammatical viewpoint. When objects are in a list, the separating commas shouldn’t be considered as signifying a new clause. Although new clauses are being identified, the resulting breakdown of the sentences is adequate when used with StoryFramer’s pronominal coreferencing algorithm. Identifying additional ‘clauses’ in the case of lists doesn’t effect the coreferencing result. However, failure to identify a clause in the text could.

4.5.5 Pronominal Coreferencing Algorithm

A basic method for coreference resolution would be to find the first antecedent to a pronoun that is of a grammatically compatible type. For a male pronoun such as “He”, this would mean finding the last mentioned male character. However this isn’t always the case and more accurate results are possible when other potentially contributing factors are considered.

The correct interpretation may be dependent on a number of other contributing factors; such as context gained from another part of the story; or the specific role interactions an object might have with a verb. There are situations that can be detected based upon the type of pronoun being coreferenced and the structure of the surrounding sentence. Being able to identify such situations can result in the correct coreference being selected, or alternatively antecedents can be ruled out as possible references.

Given that the NL input is a third-person narrative synopsis (as discussed in Section 4.1), the assumption is made that for any given pronoun the object(s) being referred that are being referenced will have been mentioned by name at some point in the text *before* the pronoun appears. Thus by using a backwards search through the preceding input sentences all the objects that the pronoun references can be found.

It is important that the pronouns are resolved in the same order that they appear in the text. The reason for this is that all object references prior to the pronoun in question need to be known. An object reference can either be a named mention of an object(s), or a pronoun referencing an object(s). It is possible for a pronoun to be referencing another pronoun, resolving them in order allows for the correct coreference links to the objects be made.

The new algorithm that has been developed for this decision making process within the StoryFramer approach is shown in Algorithm 3 and is based upon other multi-sieve approaches to coreferencing, such as CoreNLP’s deterministic method [73]. A multi-sieve approach provides an ordering over the rules that are applied. Commonly rules that check for the most specific cases are applied first, gradually applying more generalised rules. This new algorithm succeeds more than a general algorithm in this context due to way that it utilises the available additional information regarding the objects that are present in the synopsis and their types.

Algorithm 3: Pronominal Coreferencing Algorithm

```
Function Main(Input):
    for s in Sentences(Input) do
        for p in Pronouns(s) do
            // Find all objects that match the pronoun's type
            objects = FindMatchingObjects(s,p)
            // Start the sieve mechanism (Sieve 1)
            Sieve1(Input,objects,p)
        end
    end

Function Sieve1(Input,objects,p):
    ApplyRule OBJECTIVE-INFINITIVE-VERB:                    // Rule 1
        if match is found then return match
    ApplyRule OBJECTIVE-AFTER-BREAK:                        // Rule 2
        if match is found then return match
    ApplyRule REFLEXIVE:                                    // Rule 3
        if match is found then return match
    ApplyRule OBJECTIVE:                                    // Rule 4
        remove unsuitable objects from the list objects
    ApplyRule AND-POSSESSIVE:                               // Rule 5
        if match is found then return match
    ApplyRule INVOLVED-IN-ACTION:                           // Rule 6
        remove unsuitable objects from the list objects
    // At this point, no match was found. So, start Sieve2.
    Sieve2(Input,objects,p)

Function Sieve2(Input,objects,p):
    ApplyRule SINGLE-MATCH:                                 // Rule 7
        if match is found then return match
    ApplyRule MULTIPLE-MATCH:                               // Rule 8
        if match is found then return match
    ApplyRule PLURAL-MULTIPLE-MATCH:                        // Rule 9
        if match is found then return match

    // At this point, no match was found. Keep executing Sieve2 on previous sentences
       until a match is found
    s = PreviousSentence(Input,p)
    if s is defined then
        objects.add(FindMatchingObjects(s,p))
        Sieve2(Input,objects,p)
    end
```

Coreference Algorithm Method

StoryFramer's algorithm utilises two sets of rules that are referred to as *Sieve1* and *Sieve2*. For each pronoun, a list of potential objects is populated with all the objects that are referenced in the same sentence before the pronoun, providing they are type compatible with the pronoun. The next step is to apply the 6 rules of *Sieve1* in order. As a result of doing so, either an object reference has been found and the search terminated, an object has been flagged as ineligible for selection, or nothing as changed.

At this stage, if an object has not been returned, the 3 rules of *Sieve2* are now applied. If no match is returned after doing so, the list of potential objects is expanded. The list is expanded by looking to the previous sentence, adding all of the objects that it references, providing they are type compatible with the pronoun as before. The rules of *Sieve2* are then applied once again, and this process of expanding the potential object list by looking back to the next previous sentence and re-applying *Sieve2* continues until a match is found.

There is no guarantee that the input has been well-written when using a synopsis that has been written by a human. It is possible that a matching object hasn't been mentioned by name before the pronoun. In this case if the search returns no match, the pronoun isn't assigned a match and the algorithm moves onto the next pronoun.

Coreference Algorithm Rules

In this section the nine rules that make up *Sieve1* and *Sieve2* will be described. These rules are based solely on the structure of the sentences, the words involved, and their types. For each rule example sentences are provided to demonstrate its application. Examples are taken from a variety of narrative synopses in order to illustrate all of the rules.

SIEVE 1

RULE 1: OBJECTIVE INFINITIVE VERB

The infinitive form is the verb in its basic form (as it would appear in the dictionary). For the definition of this rule it is assumed that the infinitive form of a verb is always preceded by the word 'to'. The rule also accepts split infinitives (constructions where adverbs are inserted between the 'to' and the root verb).

Apply Rule When: 1. The pronoun is objective; 2. The pronoun is directly preceded by a verb in the infinitive form; 3. The last referenced object is type compatible with the pronoun; 4. At least two different objects have been referenced in the same sentence *before* the pronoun.

Action: The last referenced object is returned as the match.

Example 1: `Lisa Cuddy`, the Dean of Medicine, comes looking for `House` to berate him.
(him = `House`)

RULE 2: OBJECTIVE AFTER BREAK

Apply Rule When: 1. The pronoun is objective; 2. The pronoun occurs after a sentence break and no object reference exists between the pronoun and the sentence break; 3. The last referenced object is type compatible with the pronoun; 4. At least two different objects have been referenced in the same sentence *before* the pronoun.

Action: The last referenced object is returned as the match.

Example 1: `Bagheera` speaks to `Baloo` |and| convinces him the `jungle` isn't safe for `Mowgli`.
(him = `Baloo`)

Example 2: `He` was part of a `smuggling ring`, `he` would steal the `paintings` |and| sell them.
(them = `paintings`)

RULE 3: REFLEXIVE

Apply Rule When: 1. The pronoun is reflexive; 2. Only one object reference that is type compatible with the pronoun exists *before* the pronoun in the sentence; 3. If multiple type compatible object references exist before the pronoun in the sentence, the pronoun is treated instead as objective and Rule 4 is applied.

Action: The only type compatible object reference is returned as the match.

Example 1: `Shaggy` trips over himself.
(himself = `Shaggy`)

RULE 4: OBJECTIVE

Apply Rule When: 1. The pronoun is objective.

Action: The last referenced object becomes ineligible for selection.

Example 1: `House` thinks the `patient` has a `brain tumour`, but `Wilson` asks him to take the `case`.
(him != `Wilson`)

RULE 5: AND POSSESSIVE

Apply Rule When: 1. The pronoun is possessive; 2.The pronoun is preceded by the word 'and'; 3. The last referenced object is type compatible with the pronoun.

Action: The last referenced object is returned as the match.

Example 1: `Mowgli` joins the `elephant patrol` lead by `Hathi` and his wife `Winifred`.
(his = `Hathi`)

RULE 6: INVOLVED IN AN ACTION

Apply Rule When: 1. The words in between the next named object (going forwards in the sentence), consists of at least one verb, no sentence breaks, conjunctions or nouns.

Action: The next named object becomes ineligible for selection.

Example 1: `Louie` offers to help `Mowgli` stay in the `jungle` if he will tell `Louie` how to make fire.
(he != `Louie`)

SIEVE 2

RULE 7: SINGLE MATCH

Apply Rule When: 1. The candidate objects list contains only one match.

Action: The object is returned as the match.

Example 1: `Mowgli` is playing with his `wolf siblings`.
(his = `Mowgli`)

Example 2: House thinks the patient has a brain tumour, but Wilson asks him to take the case.
The objective rule set (him != Wilson), leaving one match remaining, resulting in (him = House)

Example 3: Louie offers to help Mowgli stay in the jungle if he will tell Louie how to make fire.
The involved in an action rule set (he != Louie), leaving one match remaining, resulting in (he = Mowgli)

RULE 8: MULTIPLE MATCH

Apply Rule When: 1. They pronoun isn't a plural; 2. The candidate objects list contains more than one match.

Action: Going backwards in the text, find the last sentence break that occurred. Select the first candidate object to be referenced after this break. If no reference is found, the next sentence break back is used, and the first candidate object reference to occur after that break is selected. This process of going backwards through the sentence breaks repeats until a match is found.

Example 1: Baloo and Bagheera head home|,| content that Mowgli is happy with his own kind.
(his = Mowgli)

RULE 9: PLURAL MULTIPLE MATCH

It is possible for a plural pronoun to be referencing a group/organisation, and object plural, or multiple characters or objects.

Apply Rule When: 1. The pronoun is a plural; 2. The candidate object list contains more than one match.

Action: The selection process is the same as the 'Multiple Match' rule, with an addition. If the match returned is a singular character or object, and other different characters or objects also exist as candidates; they are all returned as matches. Associating the pronoun with multiple references.

Example 1: He informs the gang, but when they return the painting is back on the wall.
(they = Gang)

Example 2: `Velma` says the `mystery` has `her` baffled|,| and has got `Shaggy` hungry|,| asking when they can eat?
(they = `Shaggy & Velma`)

As a result of applying the pronoun coreference algorithm on the input synopsis, every pronoun of interest present in the text will now have at least one object associated to it. This is based on the assumption that the matching object(s) for all pronouns will have been mentioned by name in the input sentences at some point *before* the pronoun.

4.6 Conclusions

This chapter has described in depth the first stage of the StoryFramer approach: the preprocessing of input synopses.

The common features of narrative synopses have been discussed, resulting in the definition of input requirements and the assumptions being made. Novel methods have been presented for the three preprocessing components: object identification; object selection, disambiguation and typing; and pronominal coreference resolution. These methods demonstrate the automation of required tasks within a semi-automated approach that utilises having an author-in-the-loop.

Chapter 5

Domain Model Acquisition

This chapter details the domain model acquisition process that is applied to a preprocessed input synopsis in order to construct a narrative planning domain model (as shown in Figure 3.5 in Section 3.3). Planning information is extracted from a preprocessed input synopsis by applying information extraction techniques that identify both the events that occur during the described narrative and the objects that participate or are associated with these events. The narrative information that is extracted from a synopsis is then used for the automated construction of a planning domain model. Default narrative predicates are introduced to provide a baseline level of causality between actions and facilitate a method of narrative control. It is demonstrated that the model is sufficient to be able to reproduce the original plot. This is shown through the automated creation of a domain model that can regenerate the input narrative as a plan. The ways in which an author can interact with the system to generalise the acquired domain model that utilise the available default predicates are discussed with a view to constructing a model capable of generating new and plausible story variants.

5.1 Extraction of Planning Information

The core purpose of a synopsis is to convey a plot summary (Section 4.1). In order to achieve this the narrative events that occur need to be described. Narrative events are events that take place and have an effect on the story being told. The extraction of such information will form the basis of the planning domain model.

In this section narrative events are defined in the context of this work. The method used to identify narrative events within natural language synopses using

information extraction techniques is described. These methods utilise the NLP annotations described in Section 4.2. A method is then presented for the naming of narrative events, looking to use relevant information that is present in the surrounding sentence to add more detail to a name where it is available. In addition to identifying the narrative events, the process developed for the identification of the objects that are associated to each event is detailed. Finally, additional descriptive information regarding objects and narrative events is extracted. The method used for this task is referred to as the extraction of properties.

5.1.1 Definition of a Narrative Event

Narrative events are defined as events that are mentioned in the synopsis that have an effect on the story. Narrative events can be either: the actions of characters; or global effects, such as the weather changing. The extraction of this information will map to actions in the planning domain model.

In order to identify narrative events in the input text, a definition of what constitutes a narrative event in natural language is required. In this work a word is considered to be representing a narrative event if it is verb, with the following exceptions (definitions taken from [12]):

- Auxiliary Verbs - Auxiliary verbs (be, do, have, and their variants) come before main verbs. When this is the case, the main verb associated with the auxiliary will be considered as a narrative event.

  The knight is unmasked - In this example, the verb "is" is an auxiliary to the main verb "unmasked."

- Copula Verbs - Also known as linking verbs, copula verbs are used to provide extra information to a subject. This providing of extra information is often conveying a property and not a narrative event.

  He was part of a smuggling ring - the copula verb "was" is used to provided extra information about the subject (He).

- Modal Verbs - Modal verbs (could, might, will, etc.) have meanings connected with uncertainty and necessity. They are often used as auxiliaries to main verbs and don't convey narrative actions.

  He would steal the paintings - The modal verb "would" is used as an auxiliary to the main verb "steal."

In the same way that there is no definitive correct or incorrect interpretation as to what constitutes an object (Section 4.3.1), the same applies to the definition of narrative events. What are considered to be the narrative events can vary and is left to the author's discretion. Stative verbs, such as those describing emotion, possession, sense and thought, are examples of verbs that an author may or may not which to be represented as a narrative event. It is plausible that a change in a character's state may represent a crucial plot point that an author would like included in the resulting planning model. To maximise the options available to an author, the approach taken is to identify and extract as many verbs that could potentially represent narrative events as possible.

5.1.2 Identifying Narrative Events

StoryFramer utilises NLP techniques in order to identify possible narrative events in input synopses. The method for doing so is presented in this section alongside examples relating to the Scooby-Doo synopsis [78] used throughout this work.

Identifying Narrative Events: Method

Given an input synopsis, narrative events are identified on a sentence by sentence basis. Each sentence is broken down further into segments that represent the clauses of a sentence. This is done using the previously identified sentence breaks (Section 4.5.4). A dependency parse tree annotation representative of the entire sentence is produced, as information about the surrounding segments may be required for additional checks later on.

For each segment a dependency parse tree annotation is obtained using Stanford CoreNLP [48]. The Part-of-Speech (POS) tag of every node in the tree is then checked, searching for eligible verbs. An eligible verb is a word that: 1) has a verb (VB) POS tag; 2) does not have an auxilary (aux) or copula (cop) dependency relation with its parent; 3) is not an identified object; 4) has not already been extracted as part of another event. Eligible verbs are then considered for narrative event extraction, discussed in Section 5.1.3. This process is represented by the function *CheckEligibleVerb()* in Algorithm 4.

It is possible for words that aren't labelled as verbs on the dependency parse tree to be representing narrative events. This can only occur when a homonym has been used. This identification method is reliant on the POS tags generated

by CoreNLP. Homonyms are a source of error for CoreNLP as homonyms have multiple meanings and can often function as differing word types, e.g., a verb and a noun. The narrative event identification algorithm makes an attempt to recognise situations where a word that has been incorrectly labelled might be representing a narrative event. It is possible that a noun may be representing a narrative event when found at the root of a dependency parse tree. When this is the case the following checks (*CheckEligibleNoun()* in Algorithm 4) are used to determine whether the word should be considered for narrative event extraction: 1) the children of the word contain at least one identified character object; 2) none of the children have a copula dependency relation with the parent word; 3) the word is not an identified object. If all are true, the word is identified as a narrative event. The outline Algorithm for this process is shown in Algorithm 4.

Algorithm 4: Narrative Event Identification Algorithm

```
Function Main(Input):
    // Input is a NL sentence
    SegmentSentence (Input)
    for segment in segments do
        DPT = GetDependencyParseTree(segment)
        for node in DPT do
            CheckEligibleVerb(node)
        end
        CheckEligibleNoun(DPT.RootNode)
    end
```

Identifying Narrative Events: Example

The following sentence taken from the Scooby-Doo synopsis will be used to demonstrate narrative event identification:

A man is driving a pick-up down a road during the night |,| unaware that the suit of armour in the back has come to life |and| left his containment.

The sentence is then broken down into segments based upon the previous identified sentence breaks. As result of doing so the three segments are:

1) A man is driving a pick-up down a road during the night
2) unaware that the suit of armour in the back has come to life
3) left his containment

```
Segment 1
-> driving/VBG (root)
  -> man/NN (nsubj)
    -> A/DT (det)
  -> is/VBZ (aux)
  -> pick-up/JJ (dobj)
    -> a/DT (det)
  -> road/NN (advmod)
    -> down/IN (case)
    -> a/DT (det)
  -> night/NN (nmod:during)
    -> during/IN (case)
    -> the/DT (det)

Segment 2
-> unaware/JJ (root)
  -> come/VBN (ccomp)
    -> that/IN (mark)
    -> suit/NN (nsubj)
      -> the/DT (det)
      -> armour/NN (nmod:of)
        -> of/IN (case)
        -> back/NN (nmod:in)
          -> in/IN (case)
          -> the/DT (det)
    -> has/VBZ (aux)
    -> life/NN (nmod:to)
      -> to/TO (case)

Segment 3
-> left/VBD (root)
  -> containment/NN (dobj)
    -> his/PRP$ (nmod:poss)
```

Figure 5.1: Dependency parse trees for the example sentence segments.

Figure 5.1 shows the dependency parse tree annotations CoreNLP produces for each of the sentence segments. Starting with the first segment, every node in the tree is checked. The root node of segment 1 (driving) is tagged as a verb (/VBG) and meets the requirements for being identified as a narrative event. Continuing the search, the next verb found is (is /VBZ). This node however doesn't meet the eligibility criteria due to having an auxiliary dependency relation (aux) with its parent (driving), and therefore isn't considered to be a narrative event.

Once every node in a segment's dependency parse tree has been checked, the same process is applied to the next segment. The node (come /VBN) from segment 2 meets the criteria and is identified as a narrative event. Finally, for segment 3, (left /VBD) is also identified.

Figure 5.2 shows two more dependency parse trees for the segments: *1) He thanks them*; and *2) Shaggy wonders what they're for*. These segments provide examples of other eligibility rules in practice. Segment 1 illustrates a situation where a homonym has been assigned an incorrect POS tag by CoreNLP. The node (thanks /NNS) is tagged as a noun despite being used as a verb in this sentence. Because the node is the root of the dependency parse tree, the noun

eligibility checks are carried out to determine whether the node is representing a narrative event. One of the checks requires one of the nodes children to be a node representing a character object in the story. As a result of the pronoun coreferencing (Section 4.5) it was determined that the pronoun (He /PRP) is referencing the character Mr. Wickles, and therefore the node meets this requirement. The node meets all of the criteria and is identified as a narrative event.

Segment 2 provides an example of a verb node ('re /VBP) that isn't considered for event extraction due to having a copula (cop) relation with it's parent.

```
Segment 1                    Segment 2

-> thanks/NNS (root)         -> wonders/VBZ (root)
  -> He/PRP (nsubj)            -> Shaggy/NN (nsubj)
  -> them/PRP (dobj)           -> for/IN (ccomp)
                                 -> what/WP (dobj)
                                 -> they/PRP (nsubj)
                                 -> 're/VBP (cop)
```

Figure 5.2: Dependency parse trees for the segments: *1) He thanks them*; and *2) Shaggy wonders what they're for*.

5.1.3 Extracting Narrative Event Names

Once a word has been identified as a narrative event, the next stage is to extract an appropriate name for that event. Ideally the name extracted for a narrative event should provide enough detail about the event such that the author can easily understand what is being represented. When a candidate word has been identified as a narrative event, it reaches this stage as just a singular verb or noun on a dependency parse tree. It is often possible to extract a more detailed name that better represents the event by considering the rest of the sentence. Failing to do so could drastically confused the meaning of an event. An example of this could be a character choosing not to carry out an action, which in itself can represent a major narrative event, e.g., *"She didn't shoot him,"* should be extracted as (DIDN'T SHOOT), and not just the verb (SHOOT).

Depending on an author's preference it may also be beneficial to combine multiple verbs into one event where it is appropriate to do so. A sentence may describe a character performing two actions simultaneously, such as, *"Scooby gets*

Algorithm 5: Narrative Event Name Extraction Algorithm

```
Function ExtractEvent(DPT, node):
    // Input is a segment dependency parse tree and a node that was identified as a
       narrative event.
    name = GetModifiedName(DPT, node)
    CheckNegatives()
    CheckJointActions()
    return name
Function GetModifiedName(DPT, node):
    modifiedName = node.word // The following checks can alter the modifiedName
       variable
    for child in node.ChildList do
        CheckNounModifiers()
        CheckObjSubj()
        CheckXcomps()
        CheckCompoundPart()
        CheckAdvmod()
        return modifiedName
    end
```

annoyed, barking and giving chase." It may be preferable to extract (BARKING AND GIVING CHASE) as a singular event, rather than two separate ones.

Extracting Narrative Event Names: Method

When a node has been identified as a narrative event (Section 5.1.2), it gets passed onto the narrative event name extraction algorithm, shown in Algorithm 5. The role of this algorithm is to extract a detailed name for the identified narrative event. This is done by considering the surrounding words in the sentence and the dependency relations provided by the segment dependency parse tree.

Taking the dependency parse tree for the segment and the node that was identified as an event, a modified name is extracted based upon the relations the node has with its children. Every child is checked for the following relations and the name modified appropriately:

- Noun Modifiers (nmod:) - Nouns modifying the verb, accompanied by words such as of, from, to and in, are added to the event name along with any determiners and adjectives that appear between the verb and the noun.

- Objects/Subjects (obj/subj) - If a node is a subject or object of the verb and has not been identified as a story object, it is added to the event name in the order that they appeared in the sentence.

- Open Clausal Compliments (xcomp) - These are predictive or clausal compliments with no subject of their own. They are added to the event name in the order they appear in the sentence, including all the words in between, apart from identified story objects.

- Phrasal Verb Particle (compound:prt) - This relation signifies a word that is a particle of the verb and should be attached in the order they appear.

- Adverb Modifier (advmod) - When an adverb has been used to add more detail to the event, it is added to the event name in the order they appeared. The word that has this relation with the parent has to have an adverb POS tag (/RB), but can't be wh-adverb (/WRB) such as who and when.

It is possible for multiple modifying relations to be found for a given event node. When this occurs they are all added to the event name such that they mirror how they appeared in the sentence. Failure to do so could produce event names that are difficult to understand.

When a modified event name has been returned by the function, a check for negatives is carried out. This is done by searching for negative auxiliaries to verbs such as, didn't, as well as negative words like 'not'.

The final check looks for additional verbs that can form a joint event with the original. Conjunctions aren't always considered sentence breaks (Section 4.5.4) and may be present in a sentence segment when they are being used to join two words together. If the two words in question are verbs then it is assumed that whoever is participating in the events are doing so simultaneously, and thus a joint event can be extracted. The name of the event is extracted by taking the first verb and all the words in between it and the second verb, and then affixing the result of getting the modified name for the second verb to the end.

Extracting Narrative Event Names: Example

The same example sentence used in Section 5.1.2 will be used to demonstrate the extraction of narrative event names.

A man is driving a pick-up down a road during the night |,| unaware that the suit of armour in the back has come to life |and| left his containment.

For this sentence, three words were identified as narrative events (*DRIVING*, *COME* and *LEFT*). Upon identification the node and the dependency parse tree for the relevant segment are passed onto the narrative event name extraction method (Algorithm 5). The dependency parse trees for the example sentence segments are shown in Figure 5.3. Also highlighted are the dependency relations that have an effect on the extraction of the event names.

Starting with (*DRIVING*) in Segment 1, the children of the node (man, is, pick-up, road, night) are checked, looking for dependency relations that alter the

Segment 1

```
-> driving/VBG (root)
  -> man/NN (nsubj)
    -> A/DT (det)
  -> is/VBZ (aux)
  -> pick-up/JJ (dobj)
    -> a/DT (det)
  -> road/NN (advmod)
    -> down/IN (case)
    -> a/DT (det)
  -> night/NN (nmod:during)
    -> during/IN (case)
    -> the/DT (det)
```

Segment 2

```
-> unaware/JJ (root)
  -> come/VBN (ccomp)
    -> that/IN (mark)
    -> suit/NN (nsubj)
      -> the/DT (det)
      -> armour/NN (nmod:of)
        -> of/IN (case)
        -> back/NN (nmod:in)
          -> in/IN (case)
          -> the/DT (det)
    -> has/VBZ (aux)
    -> life/NN (nmod:to)
      -> to/TO (case)
```

Segment 3

```
-> left/VBD (root)
  -> containment/NN (dobj)
    -> his/PRP$ (nmod:poss)
```

Figure 5.3: Dependency parse trees for the example sentence segments. The resulting extracted event names are: (driving during the night), (come to life) and (left).

extracted name. The node (night /NN) has an noun modifying (nmod:during) relation with the verb. The node is therefore added to the event name, along with the modifying word (during) and any determiners or adjectives (the) children the noun may have. The result is, (driving during the night). No other children of the verb have an effect on the event name. Finally, the dependency parse tree is checked for negatives and potential joint actions but don't apply in this example. The extracted narrative event name is (*DRIVING DURING THE NIGHT*).

The same process is applied to (*COME*) in Segment 2. The child node (life /NN) has a noun modifying (nmod:to) relation with the verb and added to the event name, resulting in (come to life). No other name modifying criteria are met and thus the extracted narrative event name is (*COME TO LIFE*). This illustrates the importance of extracting detailed event names, as the differing meanings of (come) and (come to life) is a substantial one.

No name modification occurs for (*LEFT*) in Segment 3, but only because (containment /NN) has been identified as an object in the story. If this wasn't the case, the object/subject rule would apply here and the resulting name would be (*LEFT CONTAINMENT*).

Figure 5.4 shows the extraction of four narrative events that illustrate other name modification rules. Segment 1 is an example of a negative event with a dependency relation (neg). Additionally the node (eyes /NNS) meets the requirements of the object/subject rule and is also incorporated into the name. The event extracted from the identified verb (notice /VB) is (*DON'T NOTICE EYES*).

Segment 2 presents a case where two verbs are joined together (split and look). Starting with the first mentioned verb, split, all of the words between it and the second verb are added to the name. The name modification checks are then applied to the second verb, look, with the result being added to the end of the name. This results in the event name (*SPLIT UP AND LOOK FOR CLUES*).

In Segment 3, an open clausal compliment (xcomp) is found. The extracted name is formed by taking the words as they appeared in the sentence, including any words in between that aren't objects, resulting in (*BEGIN TO MOVE*).

Segment 4 showcases both a phrasal verb particle and an adverb modifier. The words are added to the name in the order that they appear in the sentence. The event extracted is (*QUICKLY FOLLOWS BEHIND*).

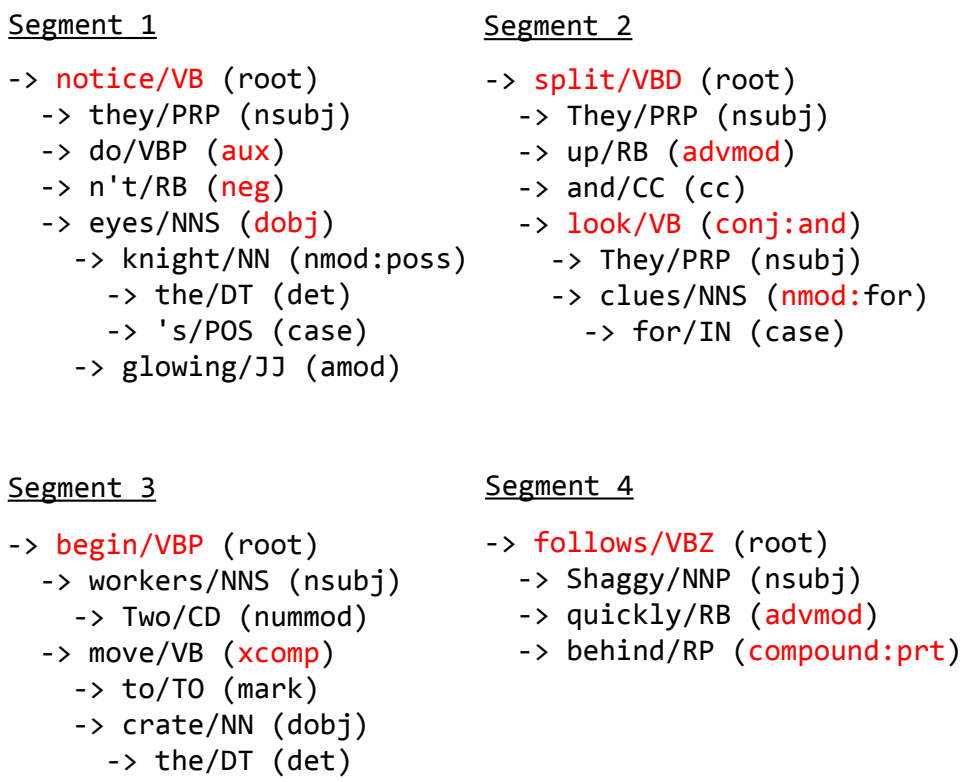

Figure 5.4: Dependency Parse Trees for the example segments: *1) They don't notice the knight's eyes glowing*; *2) They split up and look for clues*; *3) Two workers begin to move the crate*; *4) Shaggy quickly follows behind.*

5.1.4 Identifying Associated Objects

Once an event has been identified and suitably named, the next task is to identify the objects that are participating or are associated with the event. In this section the approach taken for the identification of associated objects is presented. The method selects objects to be associated with an identified event based upon the information available in the surrounding segments of the text.

Identifying Associated Objects: Method

Given a word that has been identified as a narrative event, a search of the surrounding sentence segments takes place with the aim of identifying objects that are associated with the event in question. This process comprises the following steps:

1. All objects that are referenced either by name or pronominally in the same segment as the identified event are added as associated objects.

2. The dependency parse tree for the full sentence is checked to see if the event node has any children with a subject or object dependency relation. If the subject/object child is referencing a story object(s) that isn't already associated with the event, then the object(s) is added as an associated object. The parent of the event node is also checked to see if it's a story object. If this is the case it is added as an associated object.

3. If after steps 1 and 2, no associated objects have been found, the search expands to the surrounding segments. All objects referenced in the next and previous segments are added to the list of associated objects. If still no object references were found, the search keeps checking the previous segment until an object is found or the start of the text has been reached.

4. If the event segment includes an objective pronoun (him, her, it, them) an additional check takes place. The associated objects must include an object that is not the object being referenced by the objective pronoun. If this isn't true, the objects referenced in previous segments are added until this is the case.

Algorithm 6: Identify Associated Objects Algorithm

```
Function GetAssociatedObjects(segments, eventSegment, DPT):
    // Input is the segment of the narrative event, the surrounding sentence segments and
       a dependency parse tree for the full sentence.
    AddSegmentObjectMentions(eventSegment)
    CheckSentenceDPT(DPT)
    ObjectivePronounCheck()
    if no objects found then
        CheckNeighbouringSegments()
        if no objects found then
            CheckPreviousSegments() // Previous segments are checked until
               an object mention is found or the start of the text has been reached.
        end
    end
```

Upon completion of these steps, an event will now have a list of objects (potentially empty) that have been identified as being associated with that event.

The method for identifying associated objects for a narrative event is shown in Algorithm 6.

Identifying Associated Objects: Example

The examples provided will demonstrate the various steps that form the identifying associated objects method.

1) Scooby eventually comes to a stop when he loses it|.| Shaggy doesn't have time to stop |and| trips over him.

Example 1) will focus on the two narrative events, stop and trips. For the event stop, the objects that are mentioned in the same segment (Shaggy) are added as associated objects. None of the other rules apply, the result is (*STOP - SHAGGY*). For the second event trips, the same process is applied. It has already been determined that the pronoun him is referencing the object Scooby, which gets added as an associated object. The objective pronoun rule applies in this situation, and therefore another object (not Scooby) has to be associated with the event. By adding the objects referenced in the previous segment this is achieved, resulting in (*TRIPS - SCOOBY / SHAGGY*).

2) Once freed |,| he discusses the events with the gang about there being no legend.

Example 2) uses the event freed, to demonstrate what happens when no object references are found within the same segment and to show that it can be appropriate to add references from the next segment and not just previous ones. The objects referenced in the next segment take priority as the previous segment belongs to the previous sentence. The resulting associated objects are (*FREED - HYDE WHITE / GANG / LEGEND*).

3) He thanks them |but| fears that perhaps it wasn't a good idea with Professor Hyde White disappearing.

Fears in Example 3) is an example of when associated objects are added as a result of checking the dependency parse tree for the full sentence. Figure 5.5 shows the relevant section of the full sentence dependency parse tree. As before,

```
-> He/PRP (root)                    (He = Wickles)
  -> thanks/NNS (dep)
  -> fears/VBZ (acl:relcl)
    -> them/PRP (nsubj)             (them = Gang)
    -> ,/, (punct)
    -> but/CC (advmod)
    -> idea/NN (ccomp)
      -> that/IN (mark)
      -> perhaps/RB (advmod)
...
```

Figure 5.5: The dependency parse tree used for Example 3.

the objects referenced in the same segment as the event are added as associated objects (HYDE WHITE). The fears node has a subject child (them/GANG) that isn't currently an associated object, as well as a parent (He/WICKLES) that is also an unassociated object reference. Both of these are therefore added to the list of associated objects, which results in (*FEARS - HYDE WHITE / WICKLES / GANG*).

5.1.5 Extracting Properties

Synopses aren't necessarily just a dry list of events and will often include character emotions and revelations as a method of conveying the motivations involved. Identifying such information is not necessary for extracting the plot from a synopsis. This additional descriptive information about objects and events are referred to as properties in this work. By identifying properties in a synopsis, such information can then map to predicates in the domain model. These additional predicates provide the author with more options when defining how the actions function. Given that StoryFramer is a tool that aids the creation of planning domain models, extracting additional information like this facilitates a greater level of control for the author.

Properties are identified by checking the nodes of a sentence's dependency parse tree. A node meets the requirements for being a property if: 1) The node has an adjective (JJ) POS tag; 2) The node doesn't have a subject dependency

relation with its parent node; 3) The node isn't an identified object.

Using the segment, *"Scooby gets annoyed"* as an example, the event (*GETS ANNOYED - SCOOBY*) is extracted. In addition to this the word "annoyed" meets the requirements for being a property and is identified as such. How properties can then be used will be detailed in the next section.

5.2 Automated Planning Model Construction

As discussed in Section 2.2.2, planning problems are modelled by separating the problem into two parts; a problem domain and a specific problem instance. The domain defines the world and the way in which it operates. The problem instance then defines the objects that exist in the world, the initial state, and the goal criteria for that problem instance. In this work, a problem domain and the problem instances that can be used with that domain are referred to as being parts of the same planning model.

Learning planning models is a challenging task given the amount of information required. A method is required for constructing a planning model from the limited information that has been extracted from a input narrative synopsis. This section will explain how the information extracted from a synopsis is mapped to a planning model. Default predicates that provide a mechanism for controlling the causality of actions are introduced, and it is shown that they are sufficient to be able to recreate the input plot extracted from a synopsis as a plan. The alterations that the author-in-the-loop can then make to produce a more generalised domain model are discussed in Section 5.3, which when applied can result in a model capable of the generation of new story variants.

To illustrate the processes in this section, the following short section of the Scooby-Doo example synopsis [78] will be used:

```
They split up and look for clues, not knowing that spooky
eyes in an Indian effigy are watching them. Scooby, Shaggy
and Velma bump into the Black Knight and have a brief
altercation. Scooby runs into the fossil exhibit and begins
gnawing on the bone, but is chased by the Knight. He meets up
with Shaggy and the two find one of the paintings is missing.
```

Figure 5.6: A section of the Scooby-Doo example synopsis.

5.2.1 Problem Domain Construction

A problem domain defines the world and how it operates through a set of predicates and actions. The predicates define how a world is represented. They are the relations and properties of objects that need to exist in order to capture the propositions of the state. The operators are a set of parameterised actions that describe the possible behaviours of the world. These actions consist of parameterised sets of predicates that define the actions preconditions and effects.

Problem Domain Construction: Actions

The narrative events that have been identified from an input synopsis will map to actions in the problem domain. For each narrative event, an action is created and given the same name as the narrative event. The action parameters are then added based upon the associated objects that were identified for the narrative event. A parameter is added for each associated object, the parameters are typed to match their associated object's type. For example, if a narrative event has two associated objects, (Scooby/MCHAR) and (Shaggy/MCHAR), two parameters are added to the action in the problem domain. They are then both typed as MCHAR to match the objects.

A default parameter referred to as the *StoryController* is also added. The *StoryController* is a unique object that is introduced to the planning model in order to provide a mechanism for controlling causality between actions when used alongside the default predicates. Every action in the problem domain has to include the *StoryController* object as a parameter by default. Predicates are used in conjunction with the *StoryController* object in order to capture propositions that can describe the state of the world. These propositions provide a means of recording which actions have already taken place in a plan/narrative. By including the *StoryController* as a parameter in every action, orderings can be enforced between actions, providing a method for controlling causality. The *StoryController* object has to be declared in the initial state of the problem instance that will be used alongside the problem domain.

Problem Domain Construction: Default Predicates

In an approach similar to Yordanova [92], default predicates are added to the preconditions and effects for each action. These provide a mechanism for controlling the causality between actions and the eligibility of participating objects. The predicates are named (*can-Action*) and (*has-Action*), and introduce a baseline level of causality. These predicates provide a means of tracking the actions a character has already participated in, as well as the actions that have occurred throughout the story. The control that these predicates provide is sufficient to be able to reproduce the original plot as a plan.

The final default predicate is called the object availability predicate (*available*), and is used for controlling the eligibility of objects participating in actions. This predicate works in the same way (*can-Action*) does but isn't specific to one action. An example usage of the object availability predicate would be in the case of a character dying or becoming trapped. This predicate allows an author to set a character as ineligible for any action, with the exception of actions that can resolve the state the character is currently in. This predicate can be viewed as non-essential as it isn't required for an automated regeneration of the original plot. It is however included as a default so that an author editing the domain model has access to the control that it provides.

Problem Domain Construction: Additional Candidate Predicates

The additional properties that have been extracted from an input synopsis are mapped to predicates in the problem domain. These aren't included as default in any action preconditions or effects. These are included as optional predicates that an author can utilise if they want to. Additionally a predicate is added to the problem domain for each unique object as a means of restricting the eligibility of parameters to that of specific objects if required, e.g., (*scooby ?x*). This allows for the creation of actions that can only be carried out by a specific character or object, rather than any character or object that matches the parameter type becoming an eligible parameter.

Problem Domain Construction: Example Action Mapping

Starting with the first sentence segment of the example section, "*They split up and look for clues*." The narrative event (*SPLIT UP AND LOOK FOR CLUES*) was identified along with the associated object (*They* = *GANG*). This information is now mapped to a PDDL action that is given the same name as the narrative event (SplitUpAndLookForClues).

(**:action** SplitUpAndLookForClues
:parameters()
:precondition()
:effect())

Once the PDDL has been created, the parameters can be added based upon the typing of the associated objects. The object (*GANG*) is typed as a GROUP in this example and therefore a parameter also of type GROUP is added to the action. Additionally the default StoryController object that is included in every action is added to provide a means of controlling causality between actions.

(**:action** SplitUpAndLookForClues
:parameters(?g - group ?sc - storycontroller)
:precondition()
:effect())

Default predicates can now be added to the action preconditions and effects. A (*can-Action ?x*) is added to the preconditions for each parameter so that every object participating in the action is eligible to do so. Additionally to ensure this is the case, an (*available ?x*) predicate is included for every parameter that isn't the StoryController. (*has-Action*) predicates are then added to the effects for each parameter. By doing so it is possible to track if an action has taken place, or if a character has been involved in a specific action.

(**:action** SplitUpAndLookForClues
:parameters(?g - group ?sc - storycontroller)
:precondition(and (can-SplitUpAndLookForClues ?g)
(available ?g) (can-SplitUpAndLookForClues ?sc))
:effect(and (has-SplitUpAndLookForClues ?g))
(has-SplitUpAndLookForClues ?sc)))

5.2.2 Automated Regeneration of the Input Plot

One of the aims of this work is to be able to regenerate the plot that is described by an input synopsis using the planning model that is automatically created from the extracted narrative information. Being able to recreate the original plot as a plan shows that the planning model acquired from the extracted information is representative of the input text.

The default (*can-Action*) and (*has-Action*) predicates when used alongside the StoryController object provide a mechanism for controlling the causality between actions. It is possible to reproduce the input plot as a plan using the acquired planning model, with no further input required by an author.

Figure 5.7: The identified narrative events and object associations for the example section.

Narrative events are extracted in the order that they are mentioned in the synopsis. The generated plan that conveys the original plot should therefore maintain this order, along with associating the correct objects with each action.

Figure 5.7 shows the narrative events that were identified by StoryFramer (Section 5.1.2) for the example section, including the correct object associations for each event.

In order to reproduce the original plot as a plan. The extracted identified narrative events are mapped to actions in the problem domain specifically with this goal in mind. The mapping follows the same process described in Section 5.2.1 with some changes to the preconditions and effects of each action.

Figure 5.8 shows a possible problem domain encoding that would allow for the original plot to be reproduced as a plan. Using the (*can-Action*) default predicate in conjunction with the StoryController object, the narrative events can

1. split up and look for clues - Gang

```
(:action SplitUpAndLookForClues
 :parameters (?g – group ?sc – controller)
 :precondition(and (can-SplitUpAndLookForClues ?g)
                   (can-SplitUpAndLookForClues  ?sc))
 :effect((can-NotKnowing ?sc)))
```

2. not knowing – Gang / Spooky Eyes / Indian Effigy

```
(:action NotKnowing
 :parameters (?g – group ?op – otherp ?o – other ?sc – controller)
 :precondition(and (can-NotKnowing ?g) (can-NotKnowing ?op)
                   (can-NotKnowing ?o) (can-NotKnowing  ?sc))
 :effect((can-Watching ?sc)))
```

3. watching – Gang / Spooky Eyes / Indian Effigy

```
(:action Watching
 :parameters (?g – group ?op – otherp ?o – other ?sc – controller)
 :precondition(and (can-Watching ?g) (can-Watching ?op)
                   (can-Watching ?o) (can-Watching ?sc))
 :effect((can-Bump ?sc)))
```

Figure 5.8: A possible encoding of the problem domain for reproducing the original plot.

be ordered to match that of the input. When an action is carried out, the next action in the sequence is enabled using the StoryController object and a (*can-NextAction ?sc*) effect. Alternatively this causality could be achieved by checking to see if a previous action has happened. An action's effect would become (*has-Action ?sc*) with the inclusion of a (*has-PreviousAction ?sc*) precondition.

To reproduce the original plot, the characters and objects participating in each action also have to be the same. To ensure this is the case, all parameters have to meet a (*can-Action*) precondition. Only the associated objects that have been extracted with each action will meet these requirements in the problem instance.

Figure 5.9 shows the problem instance that is generated alongside the problem domain in order to reproduce the three actions that it represents. This can be applied to the full list of extracted actions to automatically reproduce the plot of the original input synopsis.

Problem

```
(:objects    Controller – control
             Gang – group
             SpookyEyes – otherp
             IndianEffigy - other )

  (:init (can_SplitUpAndLookForClues Controller)
         (can_SplitUpAndLookForClues Gang)

         (can_NotKnowing Gang)
         (can_NotKnowing SpookyEyes)
         (can_NotKnowing IndianEffigy)

         (can_Watching Gang)
         (can_Watching SpookyEyes)
         (can_Watching IndianEffigy))

  (:goal (can_Bump Controller))
```

Plan

1. (SplitUpAndLookForClues Controller Gang)
2. (NotKnowing Controller Gang SpookyEyes IndianEffigy)
3. (Watching Controller Gang SpookyEyes IndianEffigy)

Figure 5.9: A problem instance that when used with the domain presented in Figure 5.8 produces a plan that recreates the original plot for the Scooby-Doo example.

5.3 Generalising the Domain Model

By having an author-in-the-loop, the default generated planning model produced by StoryFramer (Section 5.2.1) can be generalised to facilitate the generation of new story variants. This section will look at a number of alterations that can be made by an author to the acquired planning model such that the generation of new story variants becomes possible. The planning model generated by StoryFramer represents the input synopsis it has been acquired from, as proven by its ability to reproduce the original plot as a plan. StoryFramer provides an author with a model that can be edited to produce a model that meets their specific requirements and is capable of fulfilling their intended goal.

5.3.1 Editing the List of Actions

The approach taken when extracting narrative events from an input synopsis is to extract as many as possible, such that the options available to an author at this stage are not restricted. This results in a problem domain containing an action for every event that took place, regardless of how minor or major they were in the context of the narrative.

When planning for narrative generation, the approach taken can be described as being either high or low level based upon the number of actions that are used to describe a narrative phase. Take for example the following sequence of actions: *(argue) -> (fight) -> (grab knife) -> (stab)*. These four actions can be viewed as describing one narrative phase, that could instead be represented with a higher level action, *(kill)*. The author decides what level of planning the problem domain should represent.

The level of the generated planning model is dependent on the input synopsis it is created from. If the synopsis described narrative phases containing many actions the resulting planning model will represent the same level of detail. If an author wants the planning model to plan at a higher level, a number of alterations can be made to the actions present in the problem domain. Actions can be combined with other actions, clustered into groups and represented by a singular action, or deleted entirely depending on how an author wants the model to be used.

Merging Actions

When StoryFramer extracts narrative events from the natural language input, an attempt is made to merge actions together where it is appropriate to do so. Often this is when two verbs are clearly grammatically joined to one another, e.g., "Barking and giving chase." An author may wish to combine actions together in situations where two events didn't meet the requires to be automatically merged.

They split up and look for clues, not knowing that spooky eyes in an Indian Effigy are watching them.

The sentence above shows a situation where an author may choose to simplify a domain model by combining two actions, (*not knowing*) and (*watching*). In addition to this, actions can be renamed if desired. For example renaming the combination of these two actions, (*unaware of being watched*) may provide the action with a name that better represents the events being described.

Clustering Actions

Another method for simplifying the domain model is to cluster actions that describe similar events in the plot. It is possible that multiple actions exist that all convey the same event in the narrative. As the input synopses are written by a human, it is often the case that synonyms are used for actions that occur regularly, in an attempt to avoid sounding repetitive. Where this has purpose from a storytelling point of view, with regards to planning, the opposite is true. Every action in a planning domain should represent a unique behaviour and modify the state of the world in a different manner. By allowing an author to group actions together based upon their behaviour, this can be achieved.

An author may wish to group multiple low level actions together if they can all be categorised as a higher level action. If a synopsis describes multiple interactions of characters conversing with each other, then it is likely that multiple verbs have been used to convey this (says, agrees, replies, asks, explains, etc.). In this situation an author may decide to cluster these actions together and represent these behaviours through one, higher level (*converse*) action.

Deleting Actions

An author also has the option to delete actions entirely from the domain model. Depending on how the planning domain is going to be used, some actions will be representing events that are considered too low of a level to be included in the model.

It is also possible that an author may wish to add new actions that do not appear in the input text. However for this work the focus will be on how the extracted list of actions can be modified to create new story variants and won't include any additional new actions.

5.3.2 Managing Parameter Restrictions

The generality of the domain can be adjusted by changing the parameter restrictions of actions. By default, actions are mapped from the extracted list of narrative events and their associated objects. The parameters of actions are typed to match the types of the associated objects (MCHAR / FCHAR / OTHER / OTHERP / GROUP). The default mapping allows for any object of the same type to be eligible for a given action. An author can change this to either restrict the parameters further or relax these restrictions and allow for more objects to participate in an action where they deem appropriate.

In some cases allowing all objects of the same type to participate in an action is too relaxed and can caused characters to perform actions that aren't believable to an audience. The Scooby-Doo domain is a good example of this, as some actions are specific to a character. One event describes Scooby gnawing on a bone; an action that wouldn't make sense for any other male character to be doing. Multiple chase sequences are also present in the text and should always require the monster character to be participating. The default mapping of the domain model includes predicates for each unique character as a method of implementing parameter restrictions for specific characters. Alternatively an author could create a new predicate to serve a specific subset of characters. There could potentially be an episode of Scooby-Doo that has multiple monsters that should all be eligible for monster related actions. Creating a new (*monster ?x*) predicate to represent this group of characters and including it as an action precondition would facilitate this behaviour.

Relaxing parameter restrictions allows for more objects to participate in an

action and therefore increases the number of possible story variants a domain model can generate. The relaxing of parameter restrictions should however be carried out in such a way that the plausibility of output plan isn't compromised. This can be done by removing the type requirements and replacing these with preconditions that allow for a larger group of objects to be eligible. For example (*character ?x*) would allow for any character to participate and remove the male or female constraint.

5.3.3 Defining the Causality Between Actions

The acquired domain model doesn't define any causality between actions, but default (*can-Action*) and (*has-Action*) predicates are provided so that such causality can be implemented. In Section 5.2.2 it was shown that the causality provided by these predicates is sufficient to be able to reproduce the original plot as a plan. In order to create a domain model capable of generating new plausible story variants, a causality between actions that allows for this needs to be defined.

Controlling the Narrative Plot

A StoryController object is added to each action as a parameter as a means of controlling the plot of a narrative. When this object is used alongside the (*has-Action ?sc*) predicate, it provides a record of all the actions that have taken place in the plan so far. An order can then be forced between actions by including a precondition that checks to see if another action has already happened, (*has-AnotherAction ?sc*).

When reproducing the original plot, a strict order is applied between the actions in order to replicate the input. Doing so limits the number of possible story variants to just that of the original. In order to construct a domain model capable of generating multiple plausible story variants, a more relaxed ordering of actions is required that uses causality to ensure the final narrative is plausible.

Causality should be applied to actions that can only occur in a given order. For example, in the Scooby-Doo example, the monster cannot be unmasked without first being caught. Figure 5.10 shows the change required to the domain model to introduce this ordering between two actions. In this model the monster can only be unmasked if it has been caught. It is however possible for other actions to occur in between these two events based upon the current definitions.

```
(:action Caught
 :parameters (?sc – controller ?g – group ?m – other)
 :precondition(and   (monster ?m) (can-Caught ?g) (can-Caught ?m))
 :effect(has-Caught ?sc))

(:action Unmask
 :parameters (?sc – controller ?g – group ?m – other ?c – mchar)
 :precondition(and   (monster ?m) (can-Unmask?g) (can-Unmask ?m)
                     (can-Unmask?c) (has-Caught ?sc))
 :effect(has-Unmask ?sc))
```

Figure 5.10: Introducing causality between actions

In order to force an action to follow another directly the default (*available ?sc*) predicate can be used. Figure 5.11 demonstrates how this can be done. By using the (*available ?sc*) predicate the state of the world can be changed into a state in which only one action is available. This state can be maintained for multiple consecutive actions until the state is reverted and various other actions become available again.

Through the use of the default predicates new narratives can be generated using the constructed domain model that represent a believable action ordering.

```
(:action Caught
 :parameters (?sc – controller ?g – group ?m – other)
 :precondition(and   (monster ?m) (can-Caught ?g) (can-Caught ?m)
                     (available ?sc))
 :effect(and    (has-Caught ?sc) (not (available ?sc))))

(:action Unmask
 :parameters (?sc – controller ?g – group ?m – other ?c – mchar)
 :precondition(and   (monster ?m) (can-Unmask?g) (can-Unmask ?m)
                     (can-Unmask?c) (not (available ?sc))(has-Caught ?sc)
                     (not (has-Unmask ?sc)))
 :effect(and    (has-Unmask ?sc) (available ?sc)))
```

Figure 5.11: Forcing an action to directly follow another.

Character Causality Between Actions

How believable a narrative is depends on more than the order in which the events occur. Causality between actions has to also be consistent on a character level. It is important to make sure that the characters and objects that are involved in an action make sense to be doing so.

The (*can-Action*) and (*has-Action*) predicates aren't restricted to only the StoryController object, they can also be used for recording previous actions that a character has been involved in and controlling their eligibility for participating in actions. By utilising these predicates character causality can be achieved between actions.

Figure 5.12 demonstrates how parameter causality can be achieved. In the example Scooby and Shaggy find some missing paintings and then go on to inform the gang of this discovery. To ensure that the characters are consistent across both actions, preconditions are included in the second action to check that the characters involved both meet the (*has-Find*) requirement. If any other characters other than Scooby and Shaggy were to inform the gang of the discovery, the narrative wouldn't make sense.

```
(:action Find
 :parameters (?sc – controller ?c1 – mchar ?c2 – mchar ?p – otherp)
 :precondition(and   (can-Find ?c1) (can-Find ?c2) (can-Find ?p))
 :effect(and    (has-Find ?sc) (has-Find ?c1) (has-Find ?c2)))

(:action Inform
 :parameters (?sc – controller ?c1 – mchar ?c2 – mchar ?g – group)
 :precondition(and   (can-Inform?c1) (can-Inform?c2) (can-Inform?g)
                     (has-Find ?sc) (has-Find ?c1) (has-Find ?c2))
 :effect(has-Inform?sc))
```

Figure 5.12: Ensuring parameter causality between actions.

Generalised Domain Example

An example of how the generated planning model can be generalised by an author can be found as part of the worked example in Section 6.7.

5.4 Conclusions

This chapter detailed the second stage of the StoryFramer approach: the domain model acquisition process that is applied to preprocessed input synopses.

Methods have been presented for the automated extraction of narrative planning information from preprocessed synopses. The method developed that automatically constructs a narrative planning model based upon the extracted information has been described.

It was shown that the acquired planning model is representative of an input synopsis by demonstrating that it is capable of automatically reproducing the original plot described by a synopsis as a plan. Alterations that can be made to generalise the default planning model by an author have been presented. These alterations utilise the default predicates and narrative control mechanisms that the model provides. Through these amendments it was shown that a model can be created that is capable of generating new story variants.

Chapter 6

Worked Example: The Jungle Book

In order to demonstrate the StoryFramer approach, this chapter presents a detailed worked example that follows the processes involved in producing a generalised narrative domain model from an input synopsis. This 'start to finish' example of StoryFramer will serve as a proof of concept. The level of interaction that is required by an author in order to facilitate the process will be thoroughly documented, illustrating how much of the process StoryFramer is able to automate. An overview of the StoryFramer approach is presented in Figure 3.5 (Section 3.3) and this chapter follows the same ordering of components. The components are: 1) Object Identification; 2) Object Disambiguation and Typing; 3) Pronominal Coreference Resolution; 4) Extraction of Planning Information; 5) Automated Planning Domain Model Construction; 6) Domain Model Generalisation.

The goal is to take a synopsis that has been sourced online and create a generalised domain model that is capable of creating new story variants. The example used is that of the Disney 1967 animated film, The Jungle Book.

6.1 The Input Synopsis

This section presents the input synopsis used throughout this chapter that described the plot of The Jungle Book. This synopsis was sourced from Wikipedia, taken from the page of the 1967 animated film [91]. The synopsis is presented sentence by sentence. The sentences are numbered, starting at 0. These numbered sentences are used throughout the chapter when discussing the processes of StoryFramer and presenting the results of each process.

6.1.1 The Jungle Book Synopsis

0. Mowgli, a young orphan boy, is found in a basket in the deep jungles of India by Bagheera, a black panther who promptly takes him to a mother wolf who has just had cubs.
1. She raises him along with her own cubs and Mowgli soon becomes well acquainted with jungle life.
2. Mowgli is shown ten years later, playing with his wolf siblings.
3. One night, when the wolf tribe learns that Shere Khan, a man-eating Bengal tiger, has returned to the jungle, they realize that Mowgli must be taken to the ”Man-Village” for his (and their) own safety.
4. Bagheera volunteers to escort him back.
5. They leave that very night, but Mowgli is determined to stay in the jungle.
6. He and Bagheera rest in a tree for the night, where Kaa, a hungry python, tries to devour Mowgli, but Bagheera intervenes.
7. The next morning, Mowgli tries to join the elephant patrol led by Colonel Hathi and his wife Winifred.
8. Bagheera finds Mowgli, but after a fight decides to leave Mowgli on his own.
9. Mowgli soon meets up with the laid-back, fun-loving bear Baloo, who promises to raise Mowgli himself and never take him back to the Man-Village.
10. Shortly afterwards, a group of monkeys kidnap Mowgli and take him to their leader, King Louie the orangutan.
11. King Louie offers to help Mowgli stay in the jungle if he will tell Louie how to make fire like other humans.
12. However, since he was not raised by humans, Mowgli does not know how to make fire.
13. Bagheera and Baloo arrive to rescue Mowgli and in the ensuing chaos, King Louie’s palace is demolished to rubble.
14. Bagheera speaks to Baloo that night and convinces him that the jungle will never be safe for Mowgli so long as Shere Khan is there.
15. In the morning, Baloo reluctantly explains to Mowgli that the Man-Village is best for the boy, but Mowgli accuses him of breaking his promise and runs away.
16. As Baloo sets off in search of Mowgli, Bagheera rallies the help of Hathi and his patrol.
17. However, Shere Khan himself, who was eavesdropping on Bagheera and Hathi’s

conversation, is now determined to hunt and kill Mowgli himself.

18. Meanwhile, Mowgli has encountered Kaa once again, but thanks to the unwitting intervention of the suspicious Shere Khan, Mowgli escapes.

19. As a storm gathers, a depressed Mowgli encounters a group of friendly vultures who accept Mowgli as a fellow outcast.

20. Shere Khan appears shortly after, scaring off the vultures and confronting Mowgli.

21. Baloo rushes to the rescue and tries to keep Shere Khan away from Mowgli, but is injured.

22. When lightning strikes a nearby tree and sets it ablaze, the vultures swoop in to distract Shere Khan while Mowgli gathers flaming branches and ties them to Shere Khan's tail.

23. Terrified of fire, the tiger panics and runs off.

24. Bagheera and Baloo take Mowgli to the edge of the Man-Village, but Mowgli is still hesitant to go there.

25. His mind soon changes when he is smitten by a beautiful young girl from the village who is coming down by the riverside to fetch water.

26. After noticing Mowgli, she "accidentally" drops her water pot.

27. Mowgli retrieves it for her and follows her into the Man-Village.

28. After Mowgli chooses to stay in the Man-Village, Baloo and Bagheera decide to head home, content that Mowgli is safe and happy with his own kind.

6.2 Object Identification

The first stage of the StoryFramer approach is the identification of the objects that appear in the input text. For each sentence, Stanford CoreNLP is used to generate a constituency parse tree annotation. Objects are identified based upon the analysis of these parse trees. The method used for the automated object identification is described fully in Section 4.3.

A visual representation of the object identification process is shown in Figure 6.1, using example sentence 0 of the Jungle Book synopsis (Section 6.1.1).

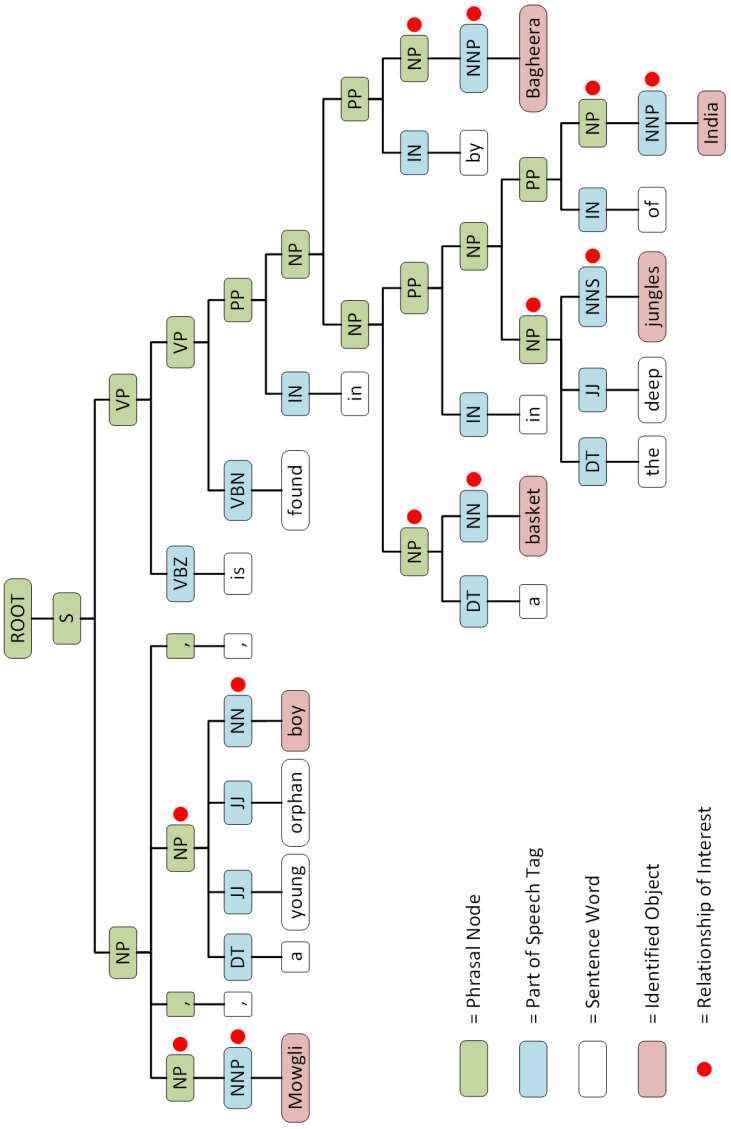

Figure 6.1: A constituency parse tree for the sentence “Mowgli, a young orphan boy, is found in a basket in the deep jungles of India by Bagheera”.

Figure 6.2 shows the results of the automated object identification for the Jungle Book synopsis. These results can be compared against a 'Gold-Standard' set of objects that have been identified by hand and are used as part of the evaluation (For further details of the Gold-Standards used to evaluate StoryFramer see Section 7.1.2 and Appendix A.2.2). Correctly identified objects that match those of the Gold-Standard are shown in black. The 10 objects highlighted in red are not considered to be objects and are referred to as additional object errors. When the results of the automated object identification are compared to that of

young orphan boy	Mowgli	deep jungles of India
basket	black panther	Bagheera
mother wolf	cubs	own cubs
jungle life	years	wolf siblings
wolf tribe	man-eating Bengal tiger	Shere Khan
jungle	Man-Village	escort
Bagheera Volunteers	night	tree
Bagheera rest	hungry python	Kaa
elephant patrol	wife Winifred	Colonel Hathi
fight	own	laid-back fun-loving bear Baloo
group of monkeys	leader	King Louie
orangutan	Louie	fire
other humans	humans	Baloo
ensuing chaos	palace	rubble
morning	boy	best
promise	search of Mowgli	help of Hathi of patrol
Hathi	conversation	eavesdropping
hunt	unwitting intervention	suspicious Shere Khan
storm	depressed Mowgli encounters	group of friendly vultures
fellow outcast	vultures	rescue
lightning	nearby tree	ablaze
ties	flaming branches	tail
tiger	edge of the Man-Village	mind
beautiful young girl	village	riverside
water	water pot	content
home	own kind	

Figure 6.2: Object Identification: The identified unique object mentions. Showing correctly identified objects (black), and additional object errors (red).

the Gold-Standard, it shows that 100% of the objects present in the text were successfully identified, with 10 additional object errors produced.

By identifying 100% of the objects mentioned in the text, an author won't be required to add any missed objects at this stage. Identifying all of the objects mentioned is the top priority for this task. Figure 6.2 shows the list of unique object mentions. These objects may have appeared multiple times within the text and it is important to correctly identify each time an object has been mentioned. The Jungle Book synopsis contains a total of 124 object mentions, 100% of which were correctly identified by StoryFramer.

The secondary goal of this task is to minimise the number of additional object errors that are produced. 10 additional object errors were produced, meaning that 7.5% of the total identified object mentions were additional object errors. When the additional object errors are analysed, the words have to be viewed in consideration with the context with which they are used. In this case 7 of the errors are describing narrative events (escort / fight / eavesdropping / hunt / intervention / rescue / ties) and 3 are adjectives (own / ablaze / content). Of the 7 narrative event errors, 6 are homonyms that can be interpreted as both nouns and verbs. When such homonyms are present in the synopses, CoreNLP tends to label these words as nouns and these additional object errors are a result of this.'Intervention' is a special case, with it being a noun that describes the act of intervening. Given the nature of the errors encountered, the additional object errors that are produced reasonable and are to be expected.

In general the names that have been extracted for each object make sense and represent an appropriate level of detail. A few mistakes can also be seen in these extracted names. A recurring error can be seen when verbs have been misinterpreted as nouns by CoreNLP. One of the object naming rules allows for a noun to be added to the identified object word, e.g., (*elephant patrol*). When words have been incorrectly labelled as nouns this can then have an effect on the naming of objects. (*Bagheera Volunteers*), (*Bagheera rest*) and (*depressed Mowgli encounters*) are all examples where this is the case. (*help of Hathi of Patrol*) is the final object that has been poorly named. The dependency relation graph for the phrase, "help of Hathi and his patrol," both 'Hathi' and 'patrol' are given a (nmod:of) relation to 'help'. The naming rule affixes an 'of' and the related word to the original noun when such a relation is found. With the rule occurring twice for the same word, it results in a poorly named object.

6.3 Object Selection, Disambiguation and Typing

When all of the possible objects in the synopsis have been identified, a selection is made based upon which objects should be represented in the planning model. Once selected, they must be disambiguated and typed before the next stages of the process. By doing so this contextual information can be leveraged for more accurate pronoun resolution results, as well as providing a means of controlling object eligibility in the planning model.

This is a task for the author. The reasoning for this is that there is no one correct answer to selecting which objects should be represented in the planning model. The selection of objects is dependent on how an author intends to use the planning model. The role of the object identification process is to identify all possible objects such to not restrict the choice an author has at this point. The objects that are selected will be referred to as story objects.

For the purpose of this example, story objects are selected from the list of identified objects based upon a simple set of requirements. An object is selected if it meets one of the following criteria:

- It represents a character or group of characters that are involved in a narrative event.
- A pronoun has been used to refer to the object.
- The correct coreferencing of a pronoun is dependent on the object being recognised as an object, e.g., The Objective Rule (Section 4.5.5).

Once story objects have been selected, they must be disambiguated. As discussed earlier in Section 4.4.2, it is possible for objects and characters in the synopsis to have multiple named references. The disambiguation of story objects refers to the identification and clustering of named mentions that are referencing the same object. As a result of this process all story objects will be represented by a unique identifier.

One or more of the following types (Section 4.4.3) are then assigned to each story object: (MCHAR / FCHAR / GROUP / OTHER / OTHERP). These types define the pronouns that can be used to reference the object, e.g., a male character (MCHAR) can be referenced using he, his and him.

Story Object	Alternative Names	Type
Mowgli	-	MCHAR
Bagheera	black panther	MCHAR
Mother wolf	-	FCHAR
Wolf siblings	-	GROUP
Wolf tribe	-	GROUP
Shere Khan	tiger	MCHAR
Kaa	-	MCHAR
Colonel Hathi	Hathi	MCHAR
Winifred	-	FCHAR
Baloo	-	MCHAR
Group of monkeys	-	GROUP
King Louie	Louie	MCHAR
Vultures	-	GROUP
Nearby tree	-	OTHER
Flaming branches	-	OTHERP
Beautiful young girl	-	FCHAR
Water pot	-	OTHER

Figure 6.3: The typed and disambiguated list of story objects for the Jungle Book example.

At this stage of processing a typed and disambiguated story object list is produced, as shown in figure 6.3. All of the objects selected keep their automatically identified names, with the exception of (*wife Winifred*), who was been renamed to (*Winifred*) for clarity. Four of the objects required disambiguation, with other named mentions being used to reference them in the synopsis. It is important to note that for the implementation of StoryFramer used in this thesis evaluation, string matching is used for the identification of objects in the text for the following automated processes. This means that if a named mention such as "laid-back fun-loving bear Baloo," contains the string of another mention used to reference the object, i.e, "Baloo," it doesn't need to be declared as an alternative name. However if an alternative mention is used that is shorter than the chosen unique identifier, e.g., "Hathi" and "Colonel Hathi," then it does need to be declared. This is a matter of author preference.

6.4 Pronominal Coreference Resolution

Now than an author has selected, typed and disambiguated the story objects, the task of pronominal coreference resolution can now be undertaken. The coreference resolution of pronouns is required because in order to extract the original plot from the synopsis, a knowledge of which characters and objects are involved in the events that occur is required. Using Algorithm 3 described in Section 4.5, the pronouns that are present in the synopsis are resolved in order using a multi-sieve rule-based approach. The rules look to find object references in the nearby sentences (antecedents) that are of a compatible type to that of the pronoun. The structure of the surrounding sentence is then used to determine the coreference where multiple antecedents are present.

The following example sentence taken from the synopsis is used to illustrate the coreferencing process for three pronouns. The story objects are highlighted in blue: (wolf tribe = GROUP), (Shere Khan & tiger = MCHAR) and (Mowgli - MCHAR). The pronouns are highlighted in red, with the superscript number representing the order in which they are coreferenced. Finally, where sentence breaks (Section 4.5.4) have been identified, square brackets have been used to signify the sentence segments/clauses. Following the example sentence, a description of how the coreference algorithm applies to each pronoun is given, detailing how the object references are selected in each situation.

> [One night,][when the wolf tribe learns that Shere Khan,][a man-eating Bengal tiger,][has returned to the jungle,][they^{1} realize that Mowgli must be taken to the "Man-Village" for his^{2} (and their^{3}) own safety.]

they^{1} - this pronoun can either be referencing a group or multiple individual objects. None of the rules in Sieve 1 apply in this situation. With no object preceding the pronoun in the same segment, the search is expanded to previous segments. tiger (an alternative reference for Shere Khan) is the first antecedent encountered. Shere Khan however isn't of a compatible type and thus the search continues. The two antecedents in the same sentence as the pronoun are wolf tribe and Shere Khan. Of these, wolf tribe is of a compatible type (GROUP) and none of the other rules state that this object cannot be the coreference; resulting in its selection. (they^{1} = wolf tribe)

his[2] - two antecedents are present in the same segment, Mowgli and wolf tribe. As it is possible for a pronoun to reference another pronoun, the result of coreferencing they[1] becomes an eligible antecedent (they[1] = wolf tribe). None of the rules in Sieve 1 apply in this situation, Sieve 2 is used for determining the coreference selection. Of Mowgli and wolf tribe, Mowgli is the only one type compatible (MCHAR) with the pronoun and no other rule states that this cannot be the coreference. (his[2] = Mowgli)

their[3] - the same two antecedents as his[2] are also considered here (Mowgli and wolf tribe). Once again, none of the rules in Sieve 1 apply here. This time it is wolf tribe that is type compatible (GROUP) with the pronoun. No rules prevent this object from being selected as the reference, resulting in its selection as the coreference. (their[3] = wolf tribe)

The pronoun coreference resolution process is applied to every sentence of the synopsis, until all pronouns have be associated with an object(s) as their coreference. The pronoun coreference resolution results for the example synopsis are shown below. The results are presented sentence-by-sentence, with the numbers correlating to the sentence numbers of the input synopsis in Section 6.1.1.

0. (0. him = Mowgli)
1. (0. she = Mother wolf) (1. him = Bagheera) (2. her = Mother wolf)
2. (0. his = Mowgli)
3. (0. they = Wolf tribe) (1. his = Mowgli) (2. their = Wolf tribe)
4. (0. him = Mowgli)
5. (0. They = Bagheera/Mowgli)
6. (0. He = Mowgli)
7. (0. his = Colonel Hathi)
8. (0. his = Mowgli)
9. (0. himself = Baloo) (1. him = Baloo)
10. (0. him = Mowgli) (their = Group of monkeys)
11. (0. he = Mowgli)
12. (0. he = King Louie)
13.
14. (0. him = Baloo)

15. (0. him = Baloo) (1. his = Mowgli)
16. (0. his = Colonel Hathi)
17. (0. himself = Shere Khan) (1. himself = Bagheera)
18.
19.
20.
21.
22. (0. it = Nearby tree) (1. them = Flaming branches)
23.
24.
25. (0. His = Mowgli) (1. he = Mowgli)
26. (0. she = Beautiful Young Girl) (1. her = Beautiful Young Girl)
27. (0. it = Water pot) (1. her = Beautiful Young Girl) (2. her = Beautiful Young Girl)
28. (0. his = Mowgli)

The Jungle Book synopsis contains a total of 35 pronouns that require coreferencing. Of these, StoryFramer correctly identifies the coreferences for 30 of the pronouns, achieving an accuracy of 85.7%. The 5 errors are shown in red. Coreference Resolution is a very complex task as determining correct coreferences can be dependent on a deep understanding of the language and the context in which it is being used. The errors produced here show examples of this.

> In the morning, Baloo reluctantly explains to Mowgli that the Man-Village is best for the boy, but Mowgli accuses him of breaking his promise and runs away.

In order to correctly coreference the pronoun his in this sentence, you would need the contextual prior knowledge of who originally made the promise, as well as an understanding of how breaking promises works, i.e., you can't break a promise made by someone else.

So that the original plot can be correctly extracted from the synopsis, an author is required to amend these errors.

6.5 Extraction of Planning Information

At this stage all of the preprocessing has been carried out and the original plot can now be identified and extracted from the input synopsis. In order to do this the narrative events that occur throughout the story and make up the plot have to be identified. Additionally the objects that participate or are involved in each event have to be identified to accurately portray the narrative being described.

Narrative events are identified by searching for words that are likely describing an action or event. NLP techniques and annotations such as part-of-speech tagging and dependency relations graphs are used to analyse each sentence and determine where narrative events should be identified. The full method for narrative event identification is described in Section 5.1.2.

The following example will illustrate how narrative events and the objects that are associated with them are identified by StoryFramer.

> [1One night,][2 when the wolf tribe learns that Shere Khan,][3 a man-eating Bengal tiger,][4 has returned to the jungle,][5 they realize that Mowgli must be taken to the "Man-Village" for his (and their) own safety.]

Figures 6.4 and 6.5 show the dependency parse trees for each segment of this example sentence. In this example all of the identified narrative events are verbs. The POS tags that indicate a verb have been highlighted in red. For each of the

```
Segment 1

-> Night/NN (root)
  -> Once/CD (dep)
  -> ,/, (punct)

Segment 2

-> learns/VBZ (root)
  -> when/WRB (advmod)
  -> tribe/NN (nsubj)
    -> the/DT (det)
    -> wolf/NN (compound)
  -> Khan/NNP (nmod:that)
    -> that/IN (case)
    -> Shere/NNP (compound)
  -> ,/, (punct)
```

```
Segment 3

-> man-eating/JJ (root)
  -> a/DT (det)
  -> Bengal/NNP (dep)
    -> tiger/NN (dep)
    -> ,/, (punct)

Segment 4

-> returned/VBN (root)
  -> has/VBZ (aux)
  -> jungle/NN (nmod:to)
    -> to/TO (case)
    -> the/DT (det)
  -> ,/, (punct)
```

Figure 6.4: Dependency parse trees for the first 4 segments of the example.

```
Segment 5
-> realize/VBP (root)                          -> -LRB-/-LRB- (punct)
  -> they/PRP (nsubj)                          -> and/CC (case)
  -> taken/VBN (ccomp)                         -> -RRB-/-RRB- (punct)
    -> that/IN (mark)                       -> safety/NN (nmod:tmod)
    -> Mowgli/NNP (nsubjpass)                 -> own/JJ (amod)
    -> must/MD (aux)                     -> ./. (punct)
    -> be/VB (auxpass)
    -> Man-Village/NNP (nmod:to)
      -> to/TO (case)
      -> the/DT (det)
      -> ``/`` (punct)
      -> ''/'' (punct)
      -> his/PRP$ (nmod:for)
        -> for/IN (case)
        -> their/PRP$ (nmod:and)
```

Figure 6.5: The dependency parse tree for segment 5 of the example.

identified words (*learns / returned / realize / taken*), the surrounding dependency relations are checked for additional information that could be included in a suitable name for the event. Doing so results in the 4 event names (*learns*), (*returned to the jungle*), (*realize*) and (*taken to the Man-Village*).

The final stage for extracting narrative events is to identify any objects that are associated with each action. This is done by seeing which objects are mentioned in the same or surrounding sentence segments.

- [2 when the wolf tribe learns that Shere Khan,]

- [3 a man-eating Bengal tiger,][4 has returned to the jungle,]

- [5 they realize that Mowgli must be taken to the “Man-Village” for his (and their) own safety.]

For the event (*learns*), both the *wolf tribe* and *Shere Khan* are mentioned by name in the same segment, resulting in (*Learns - Wolf tribe / Shere Khan*).

To find an associated object for the event (*returned to the jungle*), the previous segment is checked as no object has been mentioned in the same segment. Here *Shere Khan* has been referenced using an alternate mention, *tiger*. The event extracted is (*Returned to the jungle - Shere Khan*).

Finally for both (*realize*) and (*taken to the Man Village*), the objects referenced in the same segment are associated with these events. As a result of the previous coreferencing process the objects being referenced by the pronouns are known, (*they / their = Wolf tribe*) and (*his = Mowgli*). The events are (*Realize - Wolf tribe / Mowgli*) and (*Taken to the Man Village - Wolf tribe / Mowgli*).

As a result of applying the narrative event extraction process to every sentence in the synopsis, the following events are extracted:

0. (Found in a basket - Bagheera/Mowgli) (Promptly takes wolf - Bagheera/Mowgli/Mother wolf)
1. (Raises along with own cubs and soon becomes well acquainted - Mother wolf/Mowgli) (Well acquainted with life - Mother wolf/Mowgli)
2. (Shown later - Mowgli) (Playing - Mowgli/Wolf siblings)
3. (Learns - Wolf tribe/Shere Khan) (Returned to the jungle - Shere Khan) (Realize - Wolf tribe/-Mowgli) (Taken to the Man Village - Wolf tribe/Mowgli)
4. (Volunteers to escort back - Bagheera/Mowgli)
5. (Leave very - Bagheera/Mowgli) (Determined to stay - Mowgli)
6. (Rest in a tree - Mowgli/Bagheera) (Tries to devour - Mowgli/Kaa) (Intervenes - Bagheera/-Mowgli/Kaa)
7. (Tries to join - Mowgli/Colonel Hathi/Winifred) (Led - Mowgli/Colonel Hathi/Winifred)
8. (Finds - Bagheera/Mowgli) (Fight - Bagheera/Mowgli) (Decides to leave - Mowgli)
9. (Soon meets up - Mowgli/Baloo) (Promises to raise - Mowgli/Baloo) (Never take to the Man Village back - Mowgli/Baloo)
10. (Kidnap - Group of monkeys/Mowgli) (Take to leader - Group of monkeys/Mowgli/King Louie)
11. (Offers to help - King Louie/Mowgli) (Stay in the jungle - King Louie/Mowgli) (Tell to make - King Louie/Mowgli)
12. (Was not raised humans - Mowgli) (Does not know - Mowgli) (Make fire - Mowgli)
13. (Arrive to rescue Bagheera/Baloo/Mowgli) (ensuing Bagheera/Baloo/Mowgli) (Palace is demolished to rubble - King Louie)
14. (Speaks - Bagheera/Baloo) (Convinces - Baloo/Mowgli/Shere Khan/Bagheera)
15. (Morning - Baloo/Mowgli) (Reluctantly explains - Baloo/Mowgli) (Accuses - Mowgli/Baloo) (Breaking promise - Mowgli/Baloo) (Runs away - Mowgli)
16. (Sets in search of - Baloo/Mowgli) (Rallies the help - Bagheera/Colonel Hathi) (Patrol - Colonel Hathi)
17. (Eavesdropping - Shere Khan/Bagheera/Colonel Hathi) (Now determined to hunt - Mowgli/Shere Khan) (Kill - Mowgli/Shere Khan)
18. (Encountered again - Mowgli/Kaa) (Escapes - Mowgli) (Thanks to the unwitting intervention - Shere Khan)
19. (Storm gathers - Mowgli/Vultures) (Encounters - Mowgli/Vultures) (Accept - Mowgli/Vul-

tures)

20. (Appears shortly - Shere Khan) (Scaring off - Vultures/Shere Khan) (Confronting - Mowgli/Shere Khan)

21. (Rushes to the rescue - Baloo) (Tries to keep - Shere Khan/Mowgli/Baloo) (Injured - Baloo)

22. (Lightning strikes - Nearby Tree) (Swoop - Vultures/Shere Khan/Mowgli/Flaming branches) (Distract - Vultures/Shere Khan/Mowgli/Flaming branches) (Gathers - Vultures/Shere Khan/-Mowgli/Flaming branches) (Ties to tail - Mowgli/Shere Khan)

23. (Terrified of fire - Shere Khan) (Panics and runs off - Shere Khan)

24. (Take to the edge - Bagheera/Baloo/Mowgli) (Go there - Mowgli)

25. (Soon mind soon changes - Mowgli/Beautiful young girl) (Smitten from the village - Mowgli/Beautiful young girl) (Coming down by the riverside to fetch down - Mowgli/Beautiful young girl)

26. (Noticing - Mowgli/Beautiful young girl) (Accidentally drops - Beautiful young girl/Water pot)

27. (Retrieves - Mowgli/Water pot/Beautiful young girl) (follows in the Man-Village - Beautiful young girl/Mowgli)

28. (Chooses to stay - Mowgli) (Decided to head - Baloo/Bagheera)

In total StoryFramer successfully identifies 70 of the 75 narrative events that are present in the input synopsis. This comparison is made against a "Gold-Standard" that has been identified by hand for evaluation purposes (Appendix A.2.5). The five events that weren't identified have been included in the list of events and are highlighted in red to indicate that they were missed. They are *(Rest in a tree)*, *(Fight)*, *(Runs away)*, *(Eavesdropping)* and *(Ties to tail)*. The main verbs here are all homonyms that have multiple interpretations, and in this case, they weren't all identified due to incorrect POS tagging errors. Two narrative events were identified that were deemed to not be representing events. These are referred to as additional errors and are highlighted in blue.

In addition to the identified narrative events, 135 objects were correctly associated with events throughout the synopsis. The 8 object association errors that occurred have also been highlighted in red. An object association error can either represent: a missed object association; an incorrect object association; or both, i.e., it should be swapped for another object.

Once errors regarding the identified narrative events have been rectified by an author, the result is a list of narrative events that represents the plot of The Jungle Book. This can be used for the automated construction of a planning model.

6.6 Automated Planning Domain Model Construction

In this work the list of narrative events that has been extracted from the input and represents the plot of the synopsis is referred to as the plan trace. Using the plan trace that has been extracted by StoryFramer, a default planning model can now be constructed. The default planning model represents a basic mapping of the actions and predicates that introduces methods for controlling the generation of narratives. The goal of the default planning model is to fully represent the original plot that it has been created from; provide mechanisms for controlling narrative generation; and to not restrict the potential uses of the model. From this default model an author can then make adaptations that result in a more generalised model that meets their requirements and is capable of generating new story variants. StoryFramer uses PDDL for defining the planning problem that is separated into two parts: the problem domain; and the problem instance. This process is described fully in Section 5.2.

The Problem Domain

Each narrative event that appears in the extracted plan trace is mapped to an action in the problem domain. The actions are parameterised based upon the associated objects and their types. An additional parameter along with default predicates are added in order to introduce a baseline level of causality (Section 5.2.1). These predicates provide a means of controlling object eligibility and imposing orderings between actions.

In order to demonstrate how narrative events that appear in the plan trace are mapped to actions in the problem domain, a section of the plan trace for The Jungle Book (shown below) is used as an example. The three narrative events have been numbered, with their corresponding action mapping presented on the next page.

1. (Rest in a tree - Mowgli/Bagheera)
2. (Tries to devour - Mowgli/Kaa)
3. (Intervenes - Bagheera/Mowgli/Kaa)

1. (**:action** RestInATree
   :parameters(?c1 - mchar ?c2 - mchar ?sc - storycontroller)
   :precondition(and (can-RestInATree ?c1) (available ?c1)
   (can-RestInATree ?c2) (available ?c2) (can-RestInATree ?sc))
   :effect(and (has-RestInATree ?c1) (has-RestInATree ?c2)
   (has-RestInATree ?sc)))

2. (**:action** TriesToDevour
   :parameters(?c1 - mchar ?c2 - mchar ?sc - storycontroller)
   :precondition(and (can-TriesToDevour ?c1)(available ?c1)
   (can-TriesToDevour ?c2) (available ?c2) (can-TriesToDevour ?sc))
   :effect(and (has-TriesToDevour ?c1) (has-TriesToDevour ?c2)
   (has-TriesToDevour ?sc)))

3. (**:action** Intervenes
   :parameters(?c1 - mchar ?c2 - mchar ?c3 - mchar ?sc - storycontroller)
   :precondition(and (can-Intervenes ?c1) (available ?c1)
   (can-Intervenes ?c2) (available ?c2) (can-Intervenes ?c3)
   (available ?c3) (can-Intervenes ?sc))
   :effect(and (has-Intervenes ?c1) (has-Intervenes ?c2)
   (has-Intervenes ?c3) (has-Intervenes ?sc)))

The Problem Instance

The typed story objects are included in the problem instance along with the required default StoryController object that provides a means of controlling causality between actions. In addition to this it is assumed that the associated objects for each narrative event in the extracted plan trace are going to be given eligibility for the corresponding actions in this planning instance. Although this isn't necessarily the case it is reasonable to assume that making amendments to the eligibility of objects will be easier for an author than having to include all of them by hand. These are included in the initial state using the corresponding *(can-Action ?x)* predicate for each action.

6.7 Domain Model Generalisation

As shown in Section 5.2.2 the control that the StoryController object and the (*can-Action*) and (*has-Action*) predicates introduce is sufficient to be able to generate a plan representative of the original input story. In this section the default model is extended, by application of the changes discussed in section 5.3, to create a generalised model which is able to generate new story variants. This section models having a author-in-the-loop, implementing changes to the default Jungle Book planning model in order to create a model that fulfils an example goal.

6.7.1 An Example Goal for the Generalised Planning Model

In this worked example the default planning domain will be modified with the aim of being able to fulfil an example goal. The requirements for meeting that goal are as follows:

- The planning model must use only the actions and predicates available in the default mapping. Actions can be merged with one another or removed, but no new actions can be added by the author. No new predicates can be added to the domain, restricting the narrative control to that which is facilitated by the default mapping.
- The start and the end sequences of the original plot are to remain the same.
- A generated narrative must include all 4 "dangerous encounters" that occur in the original plot. These include: both encounters with the snake *Kaa*; The monkey kidnapping; and the final confrontation with *Shere Khan*.
- Sensible orderings and character causality should be enforced between actions where it makes sense to do so. E.g., (Promises) and (Accuses of breaking promise) should occur in that order and have the same characters participate in both.

6.7.2 Meeting the Requirements of the Example Goal

Generalising the default model such that it meets the requirements of the example goal describes a task of rearranging the original plot to generate new story variants. Action ordering and character causality should be enforced throughout using only the generative control that is provided by the predicates in the model's default mapping.

In order to meet these requirements, a simple plan illustrating the narrative control that the planning model will have to implement is shown in Figure 6.6. This plan shows a reduced and simplified list of actions that represent the original plot of The Jungle Book. This list can be achieved through merging, renaming and deleting the events extracted by StoryFramer.

The key required actions have been highlighted in red, including the start and end sequences *(Found in basket)* and *(Take to edge of village)*. The remaining four actions that represent the "dangerous encounters" can occur in any order, with the exception of *(Encountered Kaa again)*, which can only occur after *(Kaa tries to devour)*. It wouldn't make sense to have the *(Encountered Kaa again)* action before his first encounter. Additionally two non-required actions should also implement a strict ordering for the same reason and these are: *(Promises)* -> *(Accuses of breaking promise)*; and *(Runs away)* -> *(Search)*.

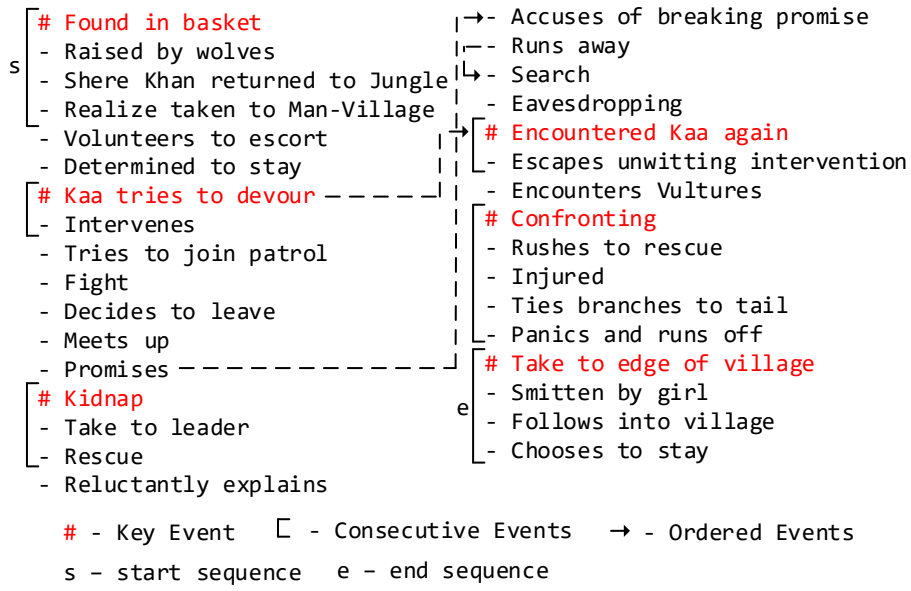

Figure 6.6: A plan illustrating how the requirements of the example goal relates to the actions in the planning domain.

Consecutive events are a more restricted version of ordered events. In addition to the events having to occur in a given order, they also have to occur immediately after one another. For every key event at least one consecutive action must occur directly afterwards. If one of the "dangerous events" such as *(Kaa tries to devour)* occurred and was not immediately resolved, the audience would assume that the danger that a character was in came to fruition. This is an entirely reasonable sequence of events and would produce a new story variant, however it would not be in keeping with the example goal for this model, as new actions cannot be created to express such scenarios. The planning model will therefore require that when an event has occurred that requires immediate resolution, a consecutive event that achieves this will follow directly afterwards.

The final consideration required for meeting the example goal is to enforce sensible character causality where possible. Given the limitations imposed by the example goal, this can be achieved by ensuring that characters participate across multiple actions where consistency is required. Using again the *(Promises)* and *(Accuses of breaking promise)* actions as an example. The characters that participate in the *(Accuses of breaking promise)* action, should also be the characters that were involved in the original *(Promises)* action. Failing to do so would result in a sequence of events that could be interpreted as a character accusing another of breaking a promise that was never made.

6.7.3 Implementing the Required Changes

By making these changes to the default planning model a generalised model will be created that meets the example goal and can generate new story variants. The different alterations that can be made to the model using only the default actions and predicates are fully described in Section 5.3.

Key Narrative Event Inclusion and Ordering

Ensuring that specific actions are included in a generated plan is simply done by declaring the corresponding *(has-Action StoryController)* in the goal state of the problem instance. To meet the key event inclusion requirements of the example goal, the goal state would be defined as follows:

```
(:goal (and     (has-FoundInBasket StoryController)
                (has-KaaTriesToDevour StoryController)
                (has-Kidnap StoryController)
                (has-EncounterKaaAgain StoryController)
                (has-Confronting StoryController)
                (has-TakeToEdgeOfVillage StoryController)))
```

The goal requirements also state that an ordering of the key events needs to be imposed. The start and end actions should mirror that of the original plot. This means that the first action should be *(FoundInBasket)* and the last actions are a sequence that starts with the key event *(TakeToEdgeOfVillage)*.

This ordering can be enforced using the default predicates in a number of ways. The simplest of which is to use the *(can-Action)* predicates. In order for the planning model to ensure that the first action is always *(FoundInBasket)* the initial state of the problem instance can be defined such that (can-FoundInBasket StoryController) is the only action available at the start of the plan. Subsequent actions can then be enabled through the effects of the *(FoundInBasket)* action. The default action can be modified as follows to meet the requirements:

```
(:action FoundInBasket
   :parameters(?c1 - mchar ?c2 - mchar ?sc - storycontroller)
   :precondition(and (can-FoundInBasket ?c1) (available ?c1)
     (can-FoundInBasket ?c2) (available ?c2) (can-FoundInBasket ?sc))
   :effect(and (has-FoundInBasket ?c1) (has-FoundInBasket ?c2)
     (has-FoundInBasket ?sc) (can-KaaTriesToDevour ?sc)
     (can-Kidnap ?sc) (can-EncounteredKaaAgain ?sc)
     (can-Confronting ?sc) (can-TakeToEdgeOfVillage ?sc)))
```

So that *(TakeToEdgeOfVillage)* is the last key narrative event to occur, the preconditions for the action can be modified such that all of the other required events need to have taken place before the conditions can be met.

(**:action** TakeToEdgeOfVillage
:parameters(?c1 - mchar ?c2 - mchar ?sc - storycontroller)
:precondition(and (can-TakeToEdgeOfVillage ?c1) (available ?c1)
(can-TakeToEdgeOfVillage ?c2) (available ?c2) (has-Kidnap ?sc)
(can-TakeToEdgeOfVillage ?sc) (has-KaaTriesToDevour ?sc)
(has-EncounteredKaaAagain ?sc) (has-Confronting ?sc))
:effect(and (has-TakeToEdgeOfVillage ?c1) (has-TakeToEdgeOfVillage ?c2)
(has-TakeToEdgeOfVillage ?sc)))

By making these amendments to the default action mappings and having an initial state where *(FoundInBasket)* is the only available first action, the key narrative event requirements will be enforced by the planning model.

Enforcing Consecutive and Ordered Events

The example goal specifies that sensible action ordering should be enforced where appropriate. In order to achieve this some events must occur in a given order, or immediately after one another. Failing to do so can result in a sequence of events that the audience may perceive to be unrealistic or confusing. Two types of event ordering can be implemented using the available default predicates.

The first method of ordering events can be achieved simply by adding a requirement to the preconditions of an action that requires another event to have occurred at any point earlier in the narrative. An example of two actions in the Jungle Book domain where this would be appropriate are the *(Promises)* and *(AccusesOfBreakingPromise)* actions. For a character to accuse another of breaking a promise without the audience first seeing that a promise had been made could result in a confusing plot progression. This ordering can be achieved by making the following amendments to the *(AccusesOfBreakingPromise)* action:

(**:action** AccusesOfBreakingPromise
:parameters(?c1 - mchar ?c2 - mchar ?sc - storycontroller)
:precondition(and (can-AccusesOfBreakingPromise ?c1) (available ?c1)
(can-AccusesOfBreakingPromise ?c2) (available ?c2)
(can-AccusesOfBreakingPromise ?sc) (has-Promises ?sc))
:effect(and(has-AccusesOfBreakingPromise?c1)
(has-AccusesOfBreakingPromise ?c2)(has-AccusesOfBreakingPromise ?sc)))

The second type of ordering implemented in the planning model allows for blocks of consecutive actions to be executed immediately after one another. This ordering can be enforced using the default predicate *(Available)*. When used in conjunction with the StoryController object, it is possible to set the world into a state that forces specific actions. When *(Available StoryController)* is true, the world being represented is one where many actions are possible, with no actions being immediately required. If the world is set to state where the StoryController isn't available, only specific actions are then considered that result in the world returning to a state where the StoryController is available once again.

The three action sequence of *(Kidnap) -> (TakeToLeader) -> (Rescue)* will be used to demonstrate how consecutive actions can be implemented. It is important to note that the preconditions for every action in the domain model will now require a condition regarding the availability of the StoryController object. *(Available ?sc)* for when multiple actions are possible, *(not (Available ?sc))* for when a specific action is required.

Figure 6.7 shows how the example actions can be amended so that a consecutive ordering is enforced between them by the planning model. The first action *(Kidnap)* sets the StoryController object as unavailable. Additionally the *(can-TakeToLeader ?sc)* effect enables the next action in the sequence. *(TakeToLeader)* is given a precondition that allows it to be executed when the StoryController object is unavailable and becomes the only available action. As this action is the second action of three, the StoryController remains unavailable and the next action is enabled. Now *(Rescue)* becomes the only available action. As the final action in the sequence, the StoryController is made available once again in the action effects.

The actions that have the ability to return the world to a state where the StoryController is available are given an additional precondition. This precondition states that the action isn't repeatable by default. The reason for this is that if an unrelated event occurs that sets the StoryController as unavailable, the action cannot be used to return the StoryController to a state where it is available.

(**:action** Kidnap
:parameters(?g1 - group ?c1 - mchar ?sc - storycontroller)
:precondition(and (can-Kidnap ?g1) (available ?g1) (can-Kidnap ?c1)
(available ?c1) (can-Kidnap ?sc) (available ?sc))
:effect(and (has-Kidnap?g1) (has-Kidnap ?c1) (has-Kidnap ?sc)
(not (available ?sc)) (can-TakeToLeader ?sc)))

(**:action** TakeToLeader
:parameters(?g1 - group ?c1 - mchar ?sc - storycontroller)
:precondition(and (can-TakeToLeader ?g1) (available ?g1)
(can-TakeToLeader ?c1) (available ?c1) (can-TakeToLeader ?sc)
(not (available ?sc)) (not (has-TakeToLeader ?sc)))
:effect(and (has-TakeToLeader?g1) (has-TakeToLeader ?c1)
(has-TakeToLeader ?sc) (can-Rescue ?sc)))

(**:action** Rescue
:parameters(?g1 - group ?c1 - mchar ?c2 - mchar
?sc - storycontroller)
:precondition(and (can-Rescue ?g1) (available ?g1) (can-Rescue ?c1)
(available ?c1) (can-Rescue ?c2) (available ?c2) (can-Rescue ?sc)
(not (available ?sc)) (not (has-Rescue ?sc)))
:effect(and (has-Rescue ?g1) (has-Rescue ?c1) (has-Rescue ?c2)
(has-Rescue ?sc) (available ?sc)))

Figure 6.7: Consecutive actions example. *(Kidnap)*, *(TakeToLeader)* & *(Rescue)*.

Character Causality Across Actions

In addition to enforcing a sensible ordering between actions, the characters that are participating across actions also have to be doing so in a consistent manner that makes sense. Character causality across actions can be achieved by utilising the default *(can-Action)* and *(has-Action)* predicates available.

To demonstrate this, the actions *(RunsAway)* and *(Search)* will be used as an example. *(RunsAway)* is an action that represents one character running away. *(Search)* is an ordered action that can occur anytime afterwards, that sees another character searching for the character that ran away. For these actions to

make sense in the narrative, the character that ran away in the first action needs to appear as a parameter in the second to indicate that they are the character being searched for. Additionally the two characters involved in the second action *(Search)*, need to be different.

No changes need to be made to the default *(RunsAway)* action. By default the *(has-RunsAway)* effect is added for all of the action's parameters. This can then be used in the preconditions of *(Search)* to identify the character that ran away.

(**:action** RunsAway
:parameters(?c1 - char ?sc - storycontroller)
:precondition(and (can-RunsAway ?c1)(available ?c1)(can-RunsAway ?sc))
:effect(and (has-RunsAway ?c1) (has-RunsAway ?sc)))

(**:action** Search
:parameters(?c1 - char ?c2 - char ?sc - storycontroller)
:precondition(and (can-Search ?c1) (available ?c1) (can-Search ?c2)
(available ?c2) (can-Search ?sc) (has-Runsaway ?c2)
(not (?c1 = ?c2)))
:effect(and (has-Search ?c1) (has-Search ?c2) (has-Search ?sc)))

The *(Search)* action is amended to include the preconditions needed for ensuring character causality across the actions. Using the equality check available in PDDL, *(not (?c1 = ?c2))* makes sure that the two character parameters are not the same object.

6.7.4 Generating New Story Variants

Now that the default planning model has been generalised in accordance with an example goal, the planning model is capable of generating new story variants that fulfil said goal.

Figure 6.8 shows an example output plan that the generalised planning model is capable of generating. The plan is an example of a new story variant which can be generated using only the default predicates that StoryFramer provides.

1. (FoundInBasket - Bagheera Mowgli)
2. (RaisedByWolves - MotherWolf Mowgli)
3. (ShereKhanReturnedToJungle WolfTribe ShereKhan)
4. (RealizeTakenToManVillage WolfTribe Mowgli)
5. (DeterminedToStay Mowgli)
6. (EncountersVultures Mowgli Vultures)
7. (Confronting Mowgli ShereKhan)
8. (RushesToRescue Bagheera)
9. (Injured Bagheera)
10. (TiesBranchesToTail Vultures FlamingBranches ShereKhan)
11. (PanicsAndRunsOff ShereKhan)
12. (KaaTriesToDevour Mowgli Kaa)
13. (Intervenes Bagheera Mowgli Kaa)
14. (Promises Bagheera Mowgli)
15. (MeetsUp Mowgli Baloo)
16. (EncounteredKaaAgain Mowgli Kaa)
17. (EscapesUnwittingIntervention Baloo Mowgli Kaa)
18. (AccusesOfBreakingPromise Mowgli Bagheera)
19. (Kidnap GroupOfMonkeys Mowgli)
20. (TakeToLeader GroupOfMonkeys Mowgli KingLouie)
21. (Rescue Bagheera Baloo Mowgli)
22. (TakeToEdgeOfVillage Bagheera Baloo Mowgli)
23. (SmittenByGirl Mowgli BeautifulYoungGirl)
24. (FollowsIntoVillage Mowgli BeautifulYoungGirl)
25. (ChoosesToStay Mowgli)

Figure 6.8: An example output plan that the generalised planning model is capable of generating.

6.8 Worked Example: Conclusions

The worked example detailed in this chapter demonstrates the StoryFramer approach that is presented in this work. The goal of the example is to demonstrate the processes involved in producing a planning model from a natural language input synopsis and serve as a proof of concept.

The Jungle Book synopsis used was an example of the target input that StoryFramer has been developed for. The synopsis represents a complex natural language description of the narrative for which it represents. Characters are referenced with multiple named mentions, pronouns are used throughout that require conferencing and narrative events are described by complex multi-clause sentences.

The 'start to finish' worked example demonstrates the effectiveness of the automated processes that StoryFramer presents: achieving an object mention identification accuracy of 100%; correctly coreferencing 85.7% of the pronouns present in the text; and correctly identifying 93.3% of the narrative events that occur throughout the story. The impressiveness of these results is emphasised by the types of errors that are encountered, with the majority of errors being expected due to their heavy reliance on contextual information and a deep understanding of natural language.

The worked example demonstrates the level of interaction that is required of an author in order to obtain the correct results for each process. The tasks completed by an author throughout the StoryFramer approach are shown to require a minimal amount of domain modelling expertise, signifying the StoryFramer's suitability as a tool supported approach to help non-technical authors with the creation of narrative planning models.

It was shown that the default mapping of the planning model and the control it facilitates is sufficient to be able to generate new story variants with an author-in-the-loop generalising the model through a number of available changes.

Chapter 7

Evaluation

This chapter contains an evaluation of the presented StoryFramer approach.

The constituent components of the StoryFramer approach are individually examined, with the outputs evaluated. The aim of this evaluation is to assess the performance of the approach on: the identification of objects from multi-clause sentences (Section 7.2); the coreference resolution of pronouns with multiple references across sentences (Section 7.3); and the identification of narrative events within synopses, including the identification of objects associated with those events (Section 7.4). The evaluation will demonstrate the processing capabilities of the approach, highlighting the complexities of each process and focusing on how the approach deals with such instances.

The level of input that is required of an author in order to obtain the correct narrative information from the synopses is evaluated, with StoryFramer's aim being to minimise the interaction required to that which is necessary. A varied selection of synopses are used as the data set for this evaluation in order to demonstrate the generality of the approach, in keeping with StoryFramer's goal of using online-sourced synopses as input.

An implementation of the StoryFramer approach has been developed in a prototype system that incorporates the components that are to be evaluated. The system is written in Java and utilises the Stanford CoreNLP toolkit [48] to produce the NLP annotations that are required for the various presented methods. The automated components of the approach are called via command line with the required input information passed to the components as text files. For all of the synopses tested no component took longer than 20 seconds to output the relevant automated results.

7.1 Narrative Synopses used in the Evaluation

For the purpose of evaluating the StoryFramer approach, 10 varied synopses have been selected. These synopses (shown in Table 7.1) provide a challenging data set for StoryFramer to be evaluated on and were chosen because they: cover multiple genres; feature a wide variety of subject matter; are sourced from a number of different online sources; vary in the level of detail they contain; and feature differing writing styles, vocabulary and language.

Synopsis	Genre	Synopsis Source
Scooby-Doo (1969)	Children's Cartoon	http://scoobydoo.wikia.com
Friends (2003)	Television Sitcom	https://www.imdb.com
House (2004)	Television Medical Drama	http://house.wikia.com
The Jungle Book (1967)	Musical Adventure Film	https://en.wikipedia.org
Toy Story (1996)	Adventure Comedy Film	https://en.wikipedia.org
Titanic (1997)	Romance & Disaster Film	https://en.wikipedia.org
Merchant of Venice (~1605)	Shakespeare Play	https://www.nosweatshakespeare.com
A Christmas Carol (1843)	Dickens Novel	https://www.sparknotes.com
Lord of the Flies (1954)	Allegorical Novel	https://www.sparknotes.com
Odyssey (~675-725 BC)	Greek Epic Poem	https://www.sparknotes.com

Table 7.1: A table of the synopses used for the evaluation.

7.1.1 Features of the Synopses

Table 7.2 contains information regarding the quantifiable metrics of the data set. These metrics will form the basis of this evaluation and provide a 'Gold-Standard' by which StoryFramer can be measured against. More information regarding the 'Gold-Standard' synopsis information can be found in Section 7.1.2.

Sentences and Pronouns

The number of sentences that a synopsis contains indicates either the length of the plot being described, or the level of detail at which the story is being told. It is more important to note the average number of object mentions (4.6) and narrative events (2.5) that appear in each sentence. This shows that the majority of sentences are complex, multi-clause sentences that make for a suitably challenging evaluation data set. Pronouns are also used throughout all of the synopses. Only pronouns that require coreferencing have been included in this count.

Synopsis	Sentences	Pronouns	Object Mentions	Narrative Events
Scooby-Doo	52	79	179	116
Friends	5	5	32	18
House	28	55	100	61
The Jungle Book	29	35	124	75
Toy Story	24	40	166	90
Titanic	38	57	192	95
Merchant of Venice	28	32	111	63
A Christmas Carol	34	52	169	76
Lord of the Flies	57	51	261	152
Odyssey	39	67	212	94
Total	334	473	1546	840

Table 7.2: The metrics for each of the evaluation synopses. These represent a hand-identified 'Gold-Standard' that StoryFramer can be measured against.

Object Mentions

An object mention is when an object is referred to by name in the text. The definition of what is considered an object in this work can be found in Section 4.3.1. Characters and physical objects that are present in the story are all considered objects. Additionally abstract nouns are included in the definition as the approach taken is to identify as many objects as reasonably possible, as to not limit the choice available to the domain author. It is possible to reference any noun using a pronoun, and thus all nouns should be identified so that the following coreferencing process isn't compromised. For a word to be classed as an object, the word has to be used as a noun within the sentence it is in. Homonyms that have noun forms but aren't being used as such, are not considered objects.

Narrative Events

Narrative events are events that have an effect on the story and can be categorised into either: the actions of characters; and events not caused by caused by characters, such as the weather changing. The full definition of a narrative event can be found in Section 5.1.1. All narrative events that appear in the synopsis should be identified by StoryFramer such that the level of detail represented by an acquired planning model matches that of the input.

7.1.2 Gold-Standard Synopsis Information

In order to evaluate the automated components of StoryFramer, the correct results for each task need to first be determined to provide a 'Gold-Standard' that the components can then be compared against.

For the metrics discussed in Section 7.1.1, the definitions of what constitutes a pronoun, object mention and narrative event in this work are given. The gold-standards produced for each of the evaluation synopses are the result of identifying these metrics in the synopses by hand. Table 7.2 shows the hand-identified metric counts for the evaluation synopses.

In addition to these counts, the correct result for each task also needs to be determined. For the task of pronominal coreference resolution, the correct coreference for each pronoun has been identified by hand. Determining the correct coreference of a pronoun is a simple task assuming the sentences have been well written. If a pronoun was encountered that had an ambiguous coreference it was ignored and omitted from the evaluation. Additionally pronouns that weren't referencing objects, e.g. referencing a previous event, were also ignored and excluded from the evaluation as these coreferences were deemed unobtainable and not required in this context.

The final gold-standard required for this evaluation is the correct narrative event object associations. This information was again identified by hand and assuming the synopsis sentences are well written this is a straightforward task. Any object that isn't clearly associated with a given narrative event was ignored and omitted from the evaluation. If the correct result of a task cannot be identified by a human author then an automated approach isn't expected to either.

Examples of the 'Gold-Standard' evaluation data for two of the evaluation synopses (Scooby-Doo & The Jungle Book) is presented in Appendix A.

7.2 Object Identification

In this section the performance of the automated object identification algorithm is evaluated. Experiments have been run to determine the accuracy of the method on the evaluation synopses. An implementation of the method presented in Section 4.3 is used for this evaluation. Objects are identified using the syntactic constituency parsing and part-of-speech tagging annotations provided by the CoreNLP toolkit [48].

7.2.1 Object Identification: Results

Table 7.3 shows the object identification results for the evaluation synopses. For each synopsis the total number of identified objects is shown and compared against the 'Gold-Standard' human-identified result. A number of objects were identified that upon review were deemed to not fall under the definition of an object, these are referred to as additional object errors. The Additional % column signifies what percentage of the identified object set were additional errors.

The method presented for the identification of objects from synopses successfully identifies 99.2% of the object mentions that appear across all of the synopses tested. This shows that the method is very effective at identifying object mentions in narrative synopses.

While it is important that all objects mentioned in the text are identified, the

Synopsis	Identified	Gold-Standard	%	Additional	Additional %
Scooby-Doo	178	179	99.4	10	5.3
Friends	32	32	100	1	3.0
House	98	100	98.0	6	5.8
The Jungle Book	124	124	100	10	7.5
Toy Story	166	166	100	6	3.5
Titanic	190	192	99.0	5	2.6
Merchant of Venice	110	111	99.1	3	2.7
A Christmas Carol	168	169	99.4	8	4.5
Lord of the Flies	258	261	98.9	8	3.0
Odyssey	210	212	99.1	7	3.2
Total	1534	1546	99.2	64	4.0

Table 7.3: Results of the object identification for the evaluation synopses

goal of this automated process is to minimise the amount of author interaction that would be required to reach the gold standard for each synopsis. The perfect solution would therefore achieve 100% identification with no additional object errors, and thus require no input from an author to reach the gold standard. On average 4% of the objects identified by StoryFramer are additional object errors, meaning that for every 96 correctly identified object mentions, 4 additional object errors are incurred. By achieving a low additional object error percentage, the authorial action that is required to achieve the 'Gold-Standard' is reduced.

7.2.2 Object Identification: Discussion

Because this solution is based upon the CoreNLP syntactic constituency parsing and part-of-speech annotations, any error within the annotation can effect the object identification process. If a word is a homonym with multiple meanings, the word can be misinterpreted by CoreNLP and given an incorrect POS tag for the context of the sentence. This in turn can effect the constituent phrases that form the constituency parse tree. Such situations are the cause of the errors that are encountered. The correct interpretation of a homonym can required a deep understanding of the natural language and the context that it has been used in. In AI this task is referred to as word sense disambiguation (WSD). The difficulty of WSD is described as an AI-Complete problem [58], a problem whose difficulty is equivalent to solving the central problem of Artificial Intelligence, i.e., The Turing Test. Given the difficulty of understanding natural language, such errors are expected and somewhat inevitable.

There are two different types of object identification error that can occur: failing to identify an object; and the identification of a word that isn't an object (additional object error). When a failure to identify an object has occurred, a noun has been mislabelled as either a verb, adjective or adverb. This then often results in the word belonging to a phrase that matches the type, as opposed to a noun phrase, and causing the word to go undetected. Similarly the additional object errors are caused by the inverse of this mislabelling, other words being labelled as nouns where they shouldn't.

The errors encountered by StoryFramer during this process are to be expected given their difficulty. By only encountering errors of this sort it emphasises the effectiveness of StoryFramer's object identification algorithm.

Example Errors from the Synopses

A number of homonym errors are encountered when running the object identification process on the Scooby-Doo synopsis. Of the errors encountered, 1 object mention goes undetected and 10 additional objects are identified incorrectly.

A mention for one of the main characters (*Shaggy*) goes undetected in a sentence where his name is identified as an adjective. Every other named mention of (*Shaggy*) is correctly identified. This error will have no effect on the result of the object identification process as the output identified objects list will include (*Shaggy*) due to being identified elsewhere in the text. If an author selects (*Shaggy*) as a story object going forward, the mention that was missed during object identification will now be recognised as an object. This is because string matching is used to identify the story objects in the text once an object list has been finalised by an author.

The additional object errors in Scooby-Doo come in the form of verbs that have been mislabelled as nouns. These include (*chase / stop / trips / thanks / break / gnawing / crashes*). These errors are expected and understandable given the nature of the problem.

7.3 Pronominal Coreference Resolution

Coreference resolution is the process of finding all the expressions in a text that refer to the same entity. The named object mentions have already been identified and disambiguated, leaving only the pronouns present in the synopses in need of coreferencing. The role of the pronoun coreference resolution process is to identify the entities that the pronouns present in a synopsis are referring to.

The pronominal coreference resolution method that is presented in Section 4.5 utilises the object information that is available in the context of StoryFramer. By incorporating object type information into a multi-sieve approach to coreferencing, a significant increase in accuracy can be achieved.

7.3.1 Pronominal Coreference Resolution: Results

In order to evaluate the performance of StoryFramer's pronominal coreference resolution algorithm, the results of running the algorithm on the evaluation synopses are compared against the default coreferencing solution that is available as part of the CoreNLP toolkit.

Experimental Setup

Both the StoryFramer and CoreNLP approaches are run on the input synopses, comparing the results against the 'Gold-Standard' human identified references. An example of a coreference 'Gold-Standard' is shown in Appendix A.1.4. If the entities identified by one of the solutions matches those of the gold-standard, the pronoun is judged to have been correctly coreferenced. If multiple entities are being referenced, they all have to be identified for the pronoun to be correctly coreferenced. In the evaluation synopses, one ambiguous pronoun was found that had no clear antecedent(s). This pronoun was removed from the results as it had no clear correct answer to compare the solutions against.

Alongside an input synopsis, StoryFramer's approach uses a typed object list for the coreferencing process. These lists are created by typing and disambiguating the identified objects, following the same process an author using StoryFramer would go through. An object is selected to be a story object if: 1) it is considered to be a main object in the story; 2) a pronoun has been used to reference the object; or 3) the coreference of a pronoun is dependent on the

object being recognised as a story object. An example of a typed object list can be found in Appendix A.1.3. The output of StoryFramer's approach is a list of every pronoun present in the synopsis, each with one or more associated objects that represent the objects being referenced. These are then compared against the gold-standard to determine the results.

To obtain the results for the CoreNLP solution, Stanford CoreNLP (Version 3.8.0) was used alongside the (2017-06-09) English model. The coreNLP pipeline was setup to use the default coreferencing solution (coref). Because the CoreNLP solution outputs coreference chains that can span an entire synopsis, each sentence was coreferenced individually. When a sentence was coreferenced however, the previous 3 sentences as well as the following sentence were also included. Therefore if a pronoun was referencing an object from another sentence, such chains could be identified. References for each pronoun were then extracted from the coreference chains if they were present, and then compared against the gold-standard.

Results

Table 7.4 shows the pronoun coreferencing results for both the StoryFramer and CoreNLP approaches. The total number of pronouns that require resolving is shown for each synopsis. The number of correctly resolved pronouns is shown in (green), and incorrectly resolved (red). On average across all of the synopses, StoryFramer correctly resolves the references for 83.7% of the pronouns. The CoreNLP approach achieves an accuracy of 40%.

By utilising the object knowledge available in this context, a significant increase in the accuracy of pronoun resolution is achieved in comparison to the default CoreNLP solution. Having all the objects identified and typed beforehand greatly reduces the number of coreferencing errors. Failing to identify particular objects and their types was a main source of error for the CoreNLP solution. This is most prominently noticeable in the House synopsis, where the main character (*House*) isn't recognised as a character, the named entity recognition (NER) incorrectly labels House as an organisation. The knock on effect of this is that no pronoun that references the character is correctly identified, contributing towards the low accuracy of 16.4%. This was a common problem, especially in the narrative synopses that feature more unconventional characters such as, the animals

Synopsis	Pronouns	StoryFramer	%	CoreNLP	%
Scooby-Doo	79	64 / 15	81.0	30 / 49	38.0
Friends	5	4 / 1	80.0	1 / 4	20.0
House	55	51 / 4	92.7	9 / 46	16.4
The Jungle Book	35	30 / 5	85.7	13 / 22	37.1
Toy Story	40	25 / 15	62.5	16 / 24	40.0
Titanic	57	52 / 5	91.2	18 / 39	31.6
Merchant of Venice	32	28 / 4	87.5	12 / 20	37.5
A Christmas Carol	52	42 / 10	80.8	28 / 24	53.8
Lord of the Flies	51	45 / 6	88.2	32 / 19	62.7
Odyssey	67	55 / 12	82.1	30 / 37	44.8
Total	473	396 / 77	83.7	189 / 284	40.0

Table 7.4: The pronominal coreferencing results for the StoryFramer algorithm and the default CoreNLP coreferencing algorithm. The number of correctly coreference pronouns are shown in (green), incorrect are shown in (red).

in The Jungle Book or the toys in Toy Story. CoreNLP fared best on the Lord of the Flies synopsis achieving 62.7%. In the synopsis all the main characters (with the exception of *Piggy*) have traditional English boys names and the problem of incorrect character recognition affecting the coreferencing is minimal.

7.3.2 Pronominal Coreference Resolution: Discussion

StoryFramer's pronoun coreferencing method achieves an accuracy of 83.7%, which is a significant improvement compared to the default CoreNLP alternative. StoryFramer still encounters a number of errors and the sources of which should be identified.

Nearly all the errors encountered during the coreference of pronouns can be accredited as a contextual error. Contextual errors is the broad category being used to describe errors that are dependent on the information that is gained from the text, rather than being directly available within the text itself. The correct coreference may be dependent on the reader's understanding of the world and the characters involved in the situation that is being described. These situations illustrate the difficulty of understanding natural language, and such errors are expected.

Pronominal Coreferencing Error Example

This example has been taken from the evaluation synopses to illustrate the contextual coreferencing problems that are encountered. The example looks to resolve the pronoun 'It' in the following sentences:

> At the library they read a book that says the glasses are for jewellers, scientists and archaeologists like Professor Hyde White. It also says they're made in England.

In this example, the previous sentence contains two objects (library) and (book) that are both type compatible with the pronoun 'It.' In order to correctly select (book) as the item being referenced, a contextual knowledge of the item itself is required. When such a situation is encountered in StoryFramer the first object in the segment is chosen as the coreference, which in this case is wrong. The order in which the two items appear or which on is chosen isn't the key to correctly coreferencing however. The sentence could be rewritten as follows:

> They read a book at the library that says the glasses are for jewellers, scientists and archaeologists like Professor Hyde White. It also says they're made in England.

Both sentences convey the same information, but now if StoryFramer were to coreference the pronoun 'It,' the first object in the segment (book) would be selected, resulting in a correctly identified coreference. Although the correct coreference is identified, it has only been achieved through chance. In order to correctly coreference both sentences, an deep understanding of the language and the specific objects involved would be required.

7.4 Narrative Event Identification

In this section the identification of narrative events and the identification of their associated objects is evaluated. Experiments have been carried out to determine how accurately this narrative information is extracted from the evaluation synopses. This evaluation uses an implementation of the method presented in Section 5.1.2. The approach utilises the CoreNLP dependency parse tree annotation in order to identify the narrative events within each sentence. Once a narrative event has been identified, the objects that are associated with said event also need to be identified. A method for the identification of associated objects is presented in Section 5.1.4 that uses information available in the surrounding segments of the text.

7.4.1 Narrative Event Identification: Results

The results of the narrative event identification for the evaluation synopses are shown in Table 7.5. For each synopsis the number of identified narrative events is shown against the 'Gold-Standard' human identified events. An example of a 'Gold-Standard' for narrative events can be found in Appendix A.1.5. Across the 10 synopses, 95.7% of the narrative events present in the texts were successfully identified. This shows the method is very effective at identifying narrative events that are present in natural language synopses.

Synopsis	Identified	Gold-S.	%	Add.	Add. %	Objs	Obj Err
Scooby-Doo	112	116	96.6	19	14.5	290	21
Friends	16	18	88.9	0	0	27	3
House	59	61	96.7	3	4.8	127	5
The Jungle Book	70	75	93.3	2	2.8	135	8
Toy Story	88	90	97.8	10	10.2	194	20
Titanic	89	95	93.7	8	8.2	196	5
Merchant of Venice	62	63	98.4	2	3.1	108	9
A Christmas Carol	72	76	94.7	8	10.0	109	17
Lord of the Flies	146	152	96.1	20	12.1	239	25
Odyssey	90	94	95.7	7	7.7	173	14
Total	804	840	95.7	79	8.9	1598	127

Table 7.5: Results of the Action identification for the evaluation synopses

Narrative events that were identified but aren't considered to be events are also recorded and referred to as additional event errors. For the synopses tested, 8.9% of the total narrative events identified were additional event errors. A perfect solution would identify 100% of the events and incur no additional event errors. The goal is to minimise the number of additional event errors without compromising the identification of the 'Gold-Standard' narrative events. By keeping the number of additional errors low, the authorial changes required to achieve a 'Gold-Standard' from StoryFramer's output is minimal.

The total number of objects that have been identified as being associated with the events in each synopsis is recorded, alongside the number of errors incurred during this process. Object association errors are evaluated against a human associated gold-standard. Object association errors are defined as the number of changes that would need to be made to the results of automated association in order to get to the gold-standard set of objects. These changes are as follows: 1) An object has been missed and should be added; 2) An incorrect object has been associated and should be removed; and 3) An incorrect object has been added and should be replaced by another object. 92.1% of the associated objects have been correctly associated with their respective narrative events, meaning that only 7.9% require an author's input to correct them. Again, the goal is to minimise the level of interaction required of an author to achieve the correct results.

7.4.2 Narrative Event Identification: Discussion

As with the object identification and pronoun coreference resolution tasks; the misinterpretation of homonyms is the main source of error when identifying narrative events in natural language. The method presented for this task is dependent on the dependency parse tree annotations produced by CoreNLP. As a consequence of this, any error in the parse tree is often reflected in the results of the event identification.

For the identification of narrative events, the majority of errors came from verbs being misinterpreted incorrectly as nouns. The other common source of identification errors relate to the handling of verbs: be, do and have. These verbs are commonly used as auxiliary verbs; i.e., in conjunction with another verb. Given the example, "He has escaped," the auxiliary verb has, is ignored

by StoryFramer as the event is described by the main verb, escaped. 'Have' is also often used to convey possession when used in conjunction with a noun, e.g., "They have a dog." The event identification errors however occur when 'Have' is used in conjunction with a noun that is conveying an event, e.g., "They have a fight." Whether possession or a narrative event is being described in this situation is dependent on the the noun itself, making it a very difficult problem to overcome.

Additional identification errors are primarily a result of the inverse misinterpretation: words being incorrectly interpreted as verbs by CoreNLP. An occasion where this isn't the case is where verbs have been used by the author as a story-telling mechanism, rather than describing a plot event. An example of this are phrases that are used to describe the chronology of the events, such as, "Following this...," or "As this is going on...".

Objects are associated with actions based up if the are mentioned in the surrounding sentence segments (clauses). One of the association errors that can occur are in situations where multiple objects have been mentioned in the same segment, but associating all of the objects with the event would be incorrect. This often depends on contextual information relating to the specifics of a verb and the implied roles that characters may have. Additionally object associations can be missed if they are mentioned in a different segment to the verb. Another common source of error relates to the usage of contextual object identifiers. Sometimes object identifiers are used that don't refer to a static group of objects. The correct interpretation of these identifiers is dependent on the current situation that they are used in. An Example of such an identifier is, "The others." When used in different contexts this identifier can refer to different groups of objects, making the task of correctly resolving the references very difficult. If only used once in a specific synopsis, this problem can be addressed by treating the identifier as a static group of objects; however this is not an option if there are multiple occurrences of the identifier throughout the text.

Example Narrative Event Errors

The Scooby-Doo synopsis contains examples of the different errors that can occur during the narrative event identification process.

1) The gang return to the museum at night and break in through an upper window

StoryFramer fails to identify the action (*break in*) as a narrative event. The homonym 'break' is incorrectly labelled as a noun by CoreNLP and due to this the action is missed.

2) Scooby, Shaggy and Velma bump into the Black Knight and have a brief altercation

The sentence above illustrates an example of the word 'have' that has been used in conjunction with a noun describing an event. Here the event (*altercation*) goes unidentified due to the false assumption that a 'have' when used in conjunction with a noun is conveying possession rather than a narrative event.

3) Shaggy makes another joke about having heard of hide and seek, but not "Hyde White".

In this example, 'hide and seek' is identified as an additional event due to being labelled as verbs, although in this context the two words have been used as a noun in reference to the game.

4) They return with the rest of their group to properly examine it.

Here is an example where an object association has been missed due to the use of a contextual object identifier. In this case (They = Scooby & Shaggy), meaning that "the rest of their group" is in reference to the other members that make up "The Gang," i.e., Fred, Daphne and Velma.

7.5 Evaluation: Conclusions

In this chapter the preprocessing and information extraction processes that are used throughout the StoryFramer approach for acquiring planning models from narrative synopses have been evaluated.

The evaluation shows that StoryFramer successfully identifies 99.2% of the object mentions present in the evaluation synopses, and 95.8% of the narrative events. When the types of errors for these tasks are considered, the impressiveness of these results is emphasised. The errors encountered are expected due to the nature of the problem being faced. In order to overcome the complex problems that this task poses a deep understanding of natural language would be required.

The evaluation of the pronoun coreference resolution algorithm presented in this work and used by StoryFramer revealed a significant increase in accuracy when compared to a default CoreNLP [48] solution. StoryFramer coreferenced 83.7% of the pronouns in the evaluation synopses correctly compared to the 40% achieved using the CoreNLP method. By leveraging the object information available in this context significant performance improvements are seen. This increase in performance will result in the reduction of authorial interaction required when undertaking such a task, increasing the level of automation provided.

Once the narrative information described in a synopsis has been correctly extracted, a planning model can then be created from this information. A method for the automated regeneration of the original plot is presented in Section 5.2.2. Additionally a default planning model can be created from the extracted information with the intention of having an author-in-the-loop generalise the model. Doing so would allow for the generation of new story variants that suit an authors requirements. Methods for achieving this are presented in Section 5.3. The worked example chapter provides a proof-of-concept, presenting an example of a generalised model in Section 6.7.

Chapter 8

Conclusion

A novel approach for the acquisition of planning domain models from narrative synopses has been presented in this thesis. The approach automates a number of the required stages including: the extraction of narrative information; and the construction of planning models representative of the input synopses. This chapter features two sections: the first presents the contributions of this work with the second discussing possible future directions.

8.1 Contributions

8.1.1 StoryFramer: An Approach to Narrative Planning Model Acquisition

The overall contribution presented in this thesis is a semi-automated approach to the acquisition of narrative planning models from input synopses. An implementation of the approach was developed throughout the course of this work. The implementation of StoryFramer required the development of a number of novel techniques which were combined in the prototype to facilitate the acquisition of planning domain models from narrative synopses.

The StoryFramer approach is specifically targeted towards narrative synopses and the available contextual information is exploited where possible to improve the accuracy with which information is identified. The approach also takes advantage of having an author in-the-loop that can amend errors that would otherwise propagate throughout the processes of this challenging task. Making such amendments doesn't require additional domain modelling expertise, maintaining

the approaches accessibility towards non-technical creators.

The approach needed to be able to extract the narrative planning information being described by natural language synopses and then automatically construct narrative planning domain models that are representative of this extracted information. These tasks have been accomplished through the development of new methods that utilise having an author-in-the-loop and exploit contextual information where available.

NLP analysis tools are used in combination with contextual information to preprocess input synopses such that narrative planning information can then be extracted from them. The approach consists of methods for: the identification of object mentions; the coreference resolution of pronouns; and the identification of the narrative events that occur throughout the synopses, including the identification of objects that are associated with each event.

A method has been developed for the automated construction of planning domain models that are mapped from the extracted narrative information. A default mapping representative of the input synopses has been presented that introduces a baseline level of narrative control through the inclusion of default predicates and objects.

The worked example (Chapter 6) that presented a start-to-finish demonstration of the approach alongside the evaluation of the implemented StoryFramer approach (Chapter 7) shows that the acquisition of planning models from narrative synopses has been successfully accomplished. The exploitation of contextual information increases the accuracy with which narrative information can be extracted from synopses. It was also shown that the acquired planning models are capable of reproducing the original plots of their respective inputs in addition to the generation of new story variants.

8.1.2 Extraction of Planning Information

A novel approach to the identification and extraction of narrative information from narrative synopses is one of the contributions that has been presented in this thesis. The approach utilises NLP tools and annotations to analyse the input sentences to identify object mentions and narrative events. The extraction of narrative information is separated into two tasks: the preprocessing of input sentences and the identification of narrative information.

The approach developed automatically identifies object mentions within the text and then utilises the available author-in-the-loop to disambiguate and type objects before progressing with subsequent tasks. Doing so requires knowledge of only the story being described and doesn't introduce a requirement for domain modelling expertise.

A pronoun coreference resolution algorithm has been developed that exploits the object information available when determining the correct coreference of a pronoun. The multi-sieve algorithm managed to achieve significant performance increases over alternatively available methods.

Methods for both the identification of narrative events and the objects that are associated with them have been developed in this work. Evaluation of these methods showed consistently good performance at identifying the narrative narrative information present in the synopses.

8.1.3 Automated Planning Domain Model Construction

A contribution of this thesis is a method for the automated construction of narrative planning models that are mapped from the extracted narrative information. Methods are presented that allow either for the original plot to be reproduced as a plan, or for the construction of a default planning model that can then be generalised by an author such that it is capable of generating new story variants.

A method for mapping the extracted narrative information to a planning model has been presented and demonstrated using the Planning Domain Definition Language (PDDL). Default predicates and objects are introduced to facilitate a baseline level of narrative control.

It has been shown that the default control given to the planning model is sufficient to be able to reproduce the original plot as a plan. In doing so it also shows that the acquired model is representative of the information that it has been constructed from.

Authors can make amendments to the StoryFramer generated default planning model using the available predicates to create a more generalised model. Through these alterations it has been shown that a model capable of generating new story variants can be produced. The aims regarding the capability of the acquired narrative planning models set out for this thesis have therefore been achieved.

8.2 Future Work

In this work input texts were restricted to that of single third-person narrative synopses. Future work would look to expand the inputs that StoryFramer could acquire planning models from. Such extensions could include: first-person stories or accounts of events; descriptions of non-narrative domains; multiple synopses regarding the same domain, e.g, multiple episodes of a TV series.

For all the possible extensions mentioned, the approach stays the same, looking to identify the all of the objects and events/actions being described, in addition to resolving all object/pronoun references that are used. Modifications to the approach would be required where necessary in order to handle the differences presented by each input. In order to handle first-person synopses the coreference resolution algorithm would have to be altered to handle first-person pronouns. The object being referenced by the first person pronouns doesn't have to be explicitly stated in the text and would likely require an author's input to identify. In order to extend the approach to allow for multiple synopses the way in which they are handled would need to be defined. The narrative information could be extracted from each synopsis and combined before a planning model is constructed representative of this information. The planning model produced would be able to generate narratives that are a recombination of actions from across the multiple synopses.

Appendix A

Evaluation Synopses

A.1 Scooby-Doo

A.1.1 Synopsis Sentences

0. A man is driving a pick-up down a road during the night, unaware that the suit of armour in the back has come to life and left his containment.
1. Its eyes glow yellow from inside its helmet as it watches the driver.
2. Close by, Scooby and Shaggy are walking home, with the latter complaining that they're out so late because the former had to see Star: Dog Ranger of the North Woods, twice; Scooby is still excited from it.
3. Just then, they hear rustling from the bushes.
4. Scooby pokes his head in and when he takes it out, a frog is on his nose.
5. When it jumps off, Scooby gets annoyed, barking and giving chase.
6. Shaggy quickly follows behind.
7. Scooby eventually comes to a stop when he loses it.
8. Shaggy doesn't have time to stop and trips over him.
9. They notice the same pick-up truck from before, now abandoned.
10. When they go for a closer look, they see the lifeless suit of armour in the passenger seat.
11. The two are quickly scared off once its head falls off.
12. They return with the rest of their group to properly examine it.
13. Fred comments on why a knight's suit of armour would be out alone in the middle of the night.
14. Shaggy jokes that maybe he's out for the night.
15. Velma chides him for his joke, as Daphne wonders who it belongs to.
16. Fred reads: "Deliver to Jameson Hyde White: Prof. of Archaeology, London, England."
17. Shaggy makes another joke about having heard of hide and seek, but not "Hyde White".
18. Velma says that is an English name.
19. Daphne also finds a delivery slip reading: "Deliver to the County Museum."
20. The gang travels to the museum (now the next day) where they deliver the knight to the

museum curator, Mr. Wickles.
21. He thanks them, but fears that perhaps it wasn't a good idea with Professor Hyde White disappearing.
22. He goes on to explain about the legend of the Black Knight and how it comes to life when the moon is full.
23. Velma asks him what Professor Hyde White was doing with it (despite having already found out beforehand), and replies that the professor was delivering it to the museum all the way from England.
24. As this is going on, they don't notice the knight's glowing eyes.
25. Two workers begin to move the crate, one of them asking Mr. Wickles where to put it.
26. He tells them to put it in the medieval room.
27. As Scooby follows the workers, he finds a strange pair of glasses.
28. He picks them up, as Daphne calls him, while Fred says they're leaving.
29. While driving down town, Velma says that the mystery has her baffled, and has got Shaggy hungry, asking when they can eat?
30. Scooby pops his head up, in agreement, while still wearing the glasses he found.
31. The others notice, and realize he must have found them at the museum.
32. Shaggy wonders what they're for, with Fred suggesting they go to the library to find out.
33. At the library they read a book which says that the glasses are for jewelers, scientists, and archaeologists like Professor Hyde White.
34. It also says they're made in England.
35. These two clues indicate that something is definitely up, and the gang plan on returning to the museum to investigate.
36. The gang return to the museum at night and break in through an upper window.
37. They split up and look for clues, not knowing that spooky eyes in an Indian effigy are watching them.
38. Scooby, Shaggy and Velma bump into the Black Knight and have a brief altercation.
39. Scooby runs into the fossil exhibit and begins gnawing on the bone, but is chased by the Knight.
40. He meets up with Shaggy and the two find one of the paintings is missing.
41. He informs the gang, but when they return, the painting is back on the wall.
42. Fred, Daphne, Velma, Shaggy and Scooby follow a trail of paint to a hidden room behind a sarcophagus and find the room full of finished and unfinished paintings.
43. The Knight appears and chases the gang into the relic room, where Scooby and Shaggy hide in a World War biplane.
44. Scooby accidentally flips the power, and the plane roars to life, flying erratically around the room until it finally crashes, taking the knight down with it.
45. The Knight is unmasked as Mr. Wickles, the curator!
46. He was part of a smuggling ring; he would steal the paintings and sell them, and then paint fakes of the paintings and put them back on the wall (that explained the hidden room, the missing painting and the paint drops on the floor).
47. Mr. Wickles knew that Professor Hyde White would know that the paintings were faked, so

he kidnapped him and thought up this Black Knight ruse.
48. Professor Hyde White is later found tied up in the Indian effigy.
49. Once freed, he discusses the events with the gang about there being no legend and that Wickles just used it to cover up his mysterious disappearance, explaining he somehow got in the armor and made him disappear on the way to the museum.
50. Suddenly they see the Black Knight in the museum office.
51. He lifts up the helmet and is revealed to be Scooby-Doo; the whole gang laughs.

A.1.2 Gold-Standard Object Identification

0. man / pick-up / night / road / back / suit of armour / life / containment
1. yellow eyes / helmet / driver
2. Shaggy / Scooby / latter / former / Star / Dog Ranger of the North Woods / Scooby
3. rustling / bushes
4. Scooby / head / frog / nose
5. Scooby
6. Shaggy
7. Scooby
8. Shaggy / time
9. same pick-up truck
10. passenger seat / lifeless suit of armour
11. head
12. rest of group
13. Fred / knight / suit of armour / middle of the night
14. Shaggy / night
15. Velma / joke / Daphne
16. Fred / Jameson Hyde White / Prof. Of Archaeology / London / England
17. Shaggy / joke / Hyde White
18. Velma / English name
19. Daphne / delivery slip / County Museum
20. gang / museum / next day / knight / museum curator / Mr. Wickles
21. good idea / Professor Hyde White
22. legend of the Black Knight / life / moon
23. Velma / Professor Hyde White / professor / museum / way / England
24. knight / eyes
25. workers / crate / Mr. Wickles
26. medieval room
27. Scooby / workers / strange pair of glasses
28. Daphne / Fred
29. town / Velma / mystery / Shaggy
30. Scooby / head / glasses
31. The others / museum

32. Shaggy / Fred / library
33. library / book / glasses / jewellers / scientists / archaeologists / Professor Hyde White
34. England
35. clues / something / gang / museum
36. gang / museum / night / upper window
37. clues / spooky eyes / Indian effigy
38. Scooby / Shaggy / Velma / Black Knight
39. Scooby / fossil exhibit / bone / knight
40. Shaggy / paintings
41. gang / paintings / wall
42. Fred / Daphne / Velma / Shaggy / Scooby / trail of paint / sarcophagus / hidden room / finished and unfinished paintings
43. Knight / gang / relic room / Scooby / Shaggy / World War biplane
44. Scooby / power / plane / life / room / knight
45. Knight / Mr. Wickles / curator
46. part of a smuggling ring / paintings / fakes of the paintings / wall / hidden room / missing painting / paint / floor
47. Mr. Wickles / Professor Hyde White / paintings / Black Knight / ruse
48. Professor Hyde White / Indian effigy
49. events / gang / legend / Wickles / mysterious disappearance / armour / way / museum
50. Black Knight / museum office
51. helmet / Scooby-Doo / whole gang

A.1.3 Typed and Disambiguated Object List

Driver (man) - MCHAR
Pick-up - OTHER
Black Knight (suit of armour / knight's suit of armour / knight) - MCHAR OTHER
Containment - OTHER
Scooby (Scooby-Doo) - MCHAR
Shaggy - MCHAR
Star: Dog Ranger - OTHER
Head - OTHER
Frog - OTHER
Fred - MCHAR
Velma - FCHAR
Daphne - FCHAR
Professor Hyde White (Jameson Hyde White: Prof. of Archaeology / Hyde White / the professor) - MCHAR
Delivery Slip - OTHER
Museum (County Museum) - OTHER

Gang - GROUP
Mr. Wickles (Wickles) - MCHAR
Legend - OTHER
Workers - GROUP
Crate - OTHER
Medieval Room - OTHER
Glasses - OTHERP
Mystery - OTHER
Library - OTHER
Book - OTHER
Spooky Eyes - OTHERP
Indian Effigy - OTHER
Fossil Exhibit - OTHER
Painting - OTHER
Paintings - OTHERP
Unfinished Paintings - OTHERP
Sarcophagus - OTHER
Hidden Room - OTHER
Relic Room - OTHER
Plane (World War Biplane) - OTHER

A.1.4 Gold-Standard Pronoun Coreference

0. (0. his = Black Knight)
1. (0. its = Black Knight) (1. its = Black Knight) (2. it = Black Knight)
2. (0. they're = Scooby/Shaggy) (1. it = Star: Dog Ranger)
3. (0. they = Scooby/Shaggy)
4. (0. his = Scooby) (1. he = Scooby) (2. it = head) (3. his = Scooby)
5. (0. it = Frog)
6. -
7. (0. he = Scooby) (1. it = Frog)
8. (0. him = Scooby)
9. (0. they = Scooby/Shaggy)
10. (0. they = Scooby/Shaggy) (1. they = Scooby/Shaggy)
11. (0. The two = Scooby/Shaggy) (1. its = Black Knight)
12. (0. they = Scooby/Shaggy) (1. their = Scooby/Shaggy) (2. it = Black Knight)
13. -
14. (0. he's = Black Knight)
15. (0. him = Shaggy) (1. his = Shaggy) (2. it = Black Knight)
16. -
17. -

18. -
19. -
20. (0. they = Gang)
21. (0. He = Mr. Wickles) (1. them = Gang) (2. it = -)
22. (0. he = Mr. Wickles) (1. it = Black Knight)
23. (0. him = Mr. Wickles) (1. it = Black Knight) (2. it = Black Knight)
24. (0. they = Gang)
25. (0. them = Workers) (1. it = Crate)
26. (0. He = Mr. Wickles) (1. them = Workers) (2. it = Crate)
27. (0. he = Scooby)
28. (0. he = Scooby) (1. them = Glasses) (2. him = Scooby) (3. they're = Fred/Daphne/Scooby)
29. (0. her = Velma) (1. they = Shaggy/Velma)
30. (0. his = Scooby) (1. he = Scooby)
31. (0. he = Scooby) (1. them = Glasses)
32. (0. they're = Glasses) (1. they = Fred/Shaggy)
33. (0. they = Fred/Shaggy)
34. (0. It = Book)
35. -
36. -
37. (0. They = Gang) (1. them = Gang)
38. -
39. -
40. (0. he = Scooby) (1. the two = Scooby/Shaggy)
41. (0. he = Scooby) (1. they = Gang)
42. -
43. -
44. (0. it = Plane) (1. it = Plane)
45. -
46. (0. He = Mr. Wickles) (1. he = Mr. Wickles) (2. them = Paintings) (3. them = Paintings)
47. (0. he = Mr. Wickles) (1. Professor Hyde White)
48. -
49. (0. he = Professor Hyde White) (1. it = Legend) (2. his = Professor Hyde White) (3. he = Mr. Wickles) (4. him = Professor Hyde White)
50. (0. they = Professor Hyde White/Gang)
51. (0. He = Black Knight)

A.1.5 Gold Standard Narrative Events

0. (Driving during the night - Driver/Pick-up) (Come to life - Black Knight) (Left - Black Knight/Containment)
1. (Eyes glow yellow as it watches - Black Knight/Driver)

2. (Walking home - Scooby/Shaggy) (Complaining - Scooby/Shaggy/Star: Dog Ranger) (See - Scooby/Shaggy/Star: Dog Ranger) (Still excited - Scooby/Star: Dog Ranger)
3. (Hear rustling from the bushes - Scooby/Shaggy)
4. (Pokes - Scooby/Head) (Takes out - Scooby/Head/Frog)
5. (Jumps off - Frog) (Gets annoyed - Scooby) (Barking and giving chase - Scooby)
6. (Quickly follows behind - Shaggy)
7. (Eventually come to a stop - Scooby/Frog) (loses - Scooby/Frog)
8. (trips - Shaggy/Scooby)
9. (Notice the same truck - Scooby/Shaggy/Pick-up)
10. (Go for closer look - Scooby/Shaggy) (See - Scooby/Shaggy/Black Knight)
11. (Quickly scared off - Scooby/Shaggy/Black Knight/Head) (Falls off - Scooby/Shaggy/Black Knight/Head)
12. (Return with the rest of the group - Scooby/Shaggy) (Properly examine - Gang/Black Knight)
13. (Comments - Fred/Black Knight)
14. (Jokes - Shaggy/Black Knight)
15. (Chides for joke - Velma/Shaggy) (Wonders who it belongs to - Daphne/Black Knight)
16. (Reads - Fred) (Deliver - Professor Hyde White)
17. (Makes another joke - Shaggy) (Heard of hide and seek - Shaggy)
18. (Says - Velma)
19. (Finds - Daphne/Delivery Slip) (Deliver to the museum - Daphne/Delivery Slip)
20. (Travels - Gang/Museum/Black Knight) (Deliver to the curator - Gang/Museum/Black Knight/Mr. Wickles)
21. (Thanks - Mr. Wickles/Gang) (Fears it wasn't a good idea - Mr. Wickles/Gang/Professor Hyde White) (Disappearing - Professor Hyde White)
22. (Goes on to explain - Mr. Wickles/Legend/Black Knight) (Comes to life - Black Knight)
23. (Asks what doing with - Velma/Professor Hyde White/Black Knight/Mr. Wickles) (Already found out beforehand - Velma/Professor Hyde White/Black Knight/Mr. Wickles) (Replies delivering from England - Mr. Wickles/Velma/Professor Hyde White/Black Knight)
24. (Don't notice eyes - Gang/Black Knight)
25. (Begin to move - Workers/Crate) (Asking where to put - Workers/Crate/Mr. Wickles)
26. (Tells to put - Workers/Crate/Mr. Wickles/Medieval Room)
27. (Follows - Scooby/Workers) (Finds - Scooby/Glasses)
28. (Picks up - Scooby/Glasses) (Calls - Daphne/Scooby) (Says leaving - Fred/Daphne/Scooby)
29. (Driving - Velma) (Says baffled - Velma/Mystery) (Has got hungry - Shaggy/Mystery) (Asking when eat - Shaggy/Velma)
30. (Pops up - Scooby/Head) (Still wearing - Scooby/Glasses) (Found - Scooby/Glasses)
31. (Realize must have found - Gang/Scooby/Glasses/Museum)
32. (Wonders - Shaggy/Glasses) (Suggesting go to library to find out - Fred/Shaggy/Library)
33. (Read - Fred/Shaggy/Book/Glasses/Library) (Says - Fred/Shaggy/Book/Glasses/Library/Professor Hyde White)
34. (Says made in England - Book/Glasses)
35. (Clues indicate something is up -) (Plan on returning - Gang/Museum) (Investigate - Gang/-

Museum)
36. (Return - Gang/Museum) (Break in - Gang/Museum)
37. (Split up and look for clues - Gang) (Not knowing - Gang/Spooky Eyes/Indian Effigy) (Watching Gang/Spooky Eyes/Indian Effigy)
38. (Bump - Scooby/Shaggy/Velma/Black Knight) (Altercation - Scooby/Shaggy/Velma/Black Knight)
39. (Runs - Scooby/Fossil Exhibit) (Begins gnawing - Scooby) (Chased - Black Knight/Scooby)
40. (Meets up - Scooby/Shaggy) (Find - Scooby/Shaggy/Paintings) (Missing - Scooby/Shaggy/-Paintings)
41. (Informs - Shaggy/Gang) (Return - Gang)
42. (Follow a trail - Fred/Velma/Daphne/Shaggy/Scooby) (Find Hidden Room - Fred/Velma/-Daphne/Shaggy/Scooby/Hidden Room/Sarcophagus)
43. (Appears - Black Knight) (Chases - Gang/Relic Room/Black Knight) (Hide - Scooby/Shaggy/Plane)
44. (Accidentally flips power - Scooby) (Roars to life - Plane) (Flying erratically around the room - Plane) (Crashes - Plane) (Taking down - Black Knight/Plane)
45. (Unmasked - Mr. Wickles/Black Knight)
46. (Steal - Mr. Wickles/Paintings) (Sell - Mr. Wickles/Paintings) (Paint fakes - Mr. Wickles/Paintings) (Put back on wall - Mr. Wickles/Paintings) (Explained hidden room - Hidden Room/Paintings)
47. (Knew he would know they were faked - Mr. Wickles/Professor Hyde White/Paintings) (Kidnapped - Mr. Wickles/Professor Hyde White) (Thought up this ruse - Mr. Wickles/Black Knihgt)
48. (Found - Professor Hyde White/Indian Effigy) (Tied up - Professor Hyde White/Indian Effigy)
49. (Freed - Professor Hyde White/Gang) (Discusses the events - Professor Hyde White/Gang) (Cover up mysterious disappearance - Professor Hyde White/Mr. Wickles/Legend) (Explaining he somehow got in the armour - Mr. Wickles) (Made him disappear on the way - Professor Hyde White/Mr. Wickles/Museum)
50. (Suddenly see in the office - Professor Hyde White/Gang/Black Knight/Museum)
51. (Lifts up the helmet - Black Knight) (Revealed to be - Black Knight/Scooby) (Laughs - Gang)

A.2 The Jungle Book

A.2.1 Synopsis Sentences

0. Mowgli, a young orphan boy, is found in a basket in the deep jungles of India by Bagheera, a black panther who promptly takes him to a mother wolf who has just had cubs.
1. She raises him along with her own cubs and Mowgli soon becomes well acquainted with jungle life.
2. Mowgli is shown ten years later, playing with his wolf siblings.
3. One night, when the wolf tribe learns that Shere Khan, a man-eating Bengal tiger, has returned to the jungle, they realize that Mowgli must be taken to the ”Man-Village” for his (and their) own safety.
4. Bagheera volunteers to escort him back.
5. They leave that very night, but Mowgli is determined to stay in the jungle.
6. He and Bagheera rest in a tree for the night, where Kaa, a hungry python, tries to devour Mowgli, but Bagheera intervenes.
7. The next morning, Mowgli tries to join the elephant patrol led by Colonel Hathi and his wife Winifred.
8. Bagheera finds Mowgli, but after a fight decides to leave Mowgli on his own.
9. Mowgli soon meets up with the laid-back, fun-loving bear Baloo, who promises to raise Mowgli himself and never take him back to the Man-Village.
10. Shortly afterwards, a group of monkeys kidnap Mowgli and take him to their leader, King Louie the orangutan.
11. King Louie offers to help Mowgli stay in the jungle if he will tell Louie how to make fire like other humans.
12. However, since he was not raised by humans, Mowgli does not know how to make fire.
13. Bagheera and Baloo arrive to rescue Mowgli and in the ensuing chaos, King Louie’s palace is demolished to rubble.
14. Bagheera speaks to Baloo that night and convinces him that the jungle will never be safe for Mowgli so long as Shere Khan is there.
15. In the morning, Baloo reluctantly explains to Mowgli that the Man-Village is best for the boy, but Mowgli accuses him of breaking his promise and runs away.
16. As Baloo sets off in search of Mowgli, Bagheera rallies the help of Hathi and his patrol.
17. However, Shere Khan himself, who was eavesdropping on Bagheera and Hathi’s conversation, is now determined to hunt and kill Mowgli himself.
18. Meanwhile, Mowgli has encountered Kaa once again, but thanks to the unwitting intervention of the suspicious Shere Khan, Mowgli escapes.
19. As a storm gathers, a depressed Mowgli encounters a group of friendly vultures who accept Mowgli as a fellow outcast.
20. Shere Khan appears shortly after, scaring off the vultures and confronting Mowgli.
21. Baloo rushes to the rescue and tries to keep Shere Khan away from Mowgli, but is injured.
22. When lightning strikes a nearby tree and sets it ablaze, the vultures swoop in to distract Shere

Khan while Mowgli gathers flaming branches and ties them to Shere Khan's tail.
23. Terrified of fire, the tiger panics and runs off.
24. Bagheera and Baloo take Mowgli to the edge of the Man-Village, but Mowgli is still hesitant to go there.
25. His mind soon changes when he is smitten by a beautiful young girl from the village who is coming down by the riverside to fetch water.
26. After noticing Mowgli, she "accidentally" drops her water pot.
27. Mowgli retrieves it for her and follows her into the Man-Village.
28. After Mowgli chooses to stay in the Man-Village, Baloo and Bagheera decide to head home, content that Mowgli is safe and happy with his own kind.

A.2.2 Gold-Standard Object Identification

0. Mowgli / young orphan boy / basket / deep jungles of India / Bagheera / black panther / mother wolf / cubs
1. own cubs / Mowgli / jungle life
2. Mowgli / years / wolf siblings
3. wolf tribe / Shere Khan / man-eating Bengal tiger / jungle / Mowgli / Man-Village
4. Bagheera / head / frog / nose
5. Mowgli / jungle
6. Bagheera / tree / night / Kaa / hungry python / Mowgli
7. Mowgli / elephant patrol / Colonel Hathi / Winifred
8. Bagheera / Mowgli
9. Mowgli / laid-back fun-loving bear Baloo / Man-Village
10. group of monkeys / Mowgli / leader / King Louie / orangutan
11. King Louie / Mowgli / jungle / Louie / fire / other humans
12. humans / Mowgli / fire
13. Bagheera / Baloo / Mowgli / ensuing chaos / King Louie / palace / rubble
14. Bagheera / Baloo / night / jungle / Mowgli / Shere Khan
15. morning / Baloo / Mowgli / Man-Village / best / boy / promise
16. Baloo / Mowgli / Bagheera / Hathi / patrol
17. Shere Khan / Bagheera / Hathi / conversation / Mowgli 18. Mowgli / Kaa / suspicious Shere Khan
19. storm / depressed Mowgli / group of friendly vultures / Mowgli / fellow outcast
20. Shere Khan / vultures / Mowgli
21. Baloo / Shere Khan / Mowgli
22. lightning / nearby tree / vultures / Shere Khan / Mowgli / flaming branches / tail
23. fire / tiger
24. Bagheera / Baloo / Mowgli / edge of the Man-Village
25. mind / beautiful young girl / village / riverside / water
26. Mowgli / water pot

27. Mowgli / Man-Village
28. Mowgli / Man-Village / Baloo / Bagheera / home / own kind

A.2.3 Typed and Disambiguated Object List

Mowgli - MCHAR
Bagheera (panther) - MCHAR
Mother wolf - FCHAR
Wolf siblings - GROUP
Wolf tribe - GROUP
Shere Khan (tiger) - MCHAR
Kaa - MCHAR
Colonel Hathi (Hathi) - MCHAR
Winifred - FCHAR
Baloo - MCHAR
Monkeys - GROUP
King Louie (Louie) - MCHAR
Vultures - GROUP
Tree - OTHER
Flaming branches - OTHERP
Young girl - FCHAR
Water pot - OTHER

A.2.4 Gold-Standard Pronoun Coreference

0. (0. him = Mowgli)
1. (0. she = Mother wolf) (1. him = Mowgli) (2. her = Mother wolf)
2. (0. his = Mowgli)
3. (0. they = Wolf tribe) (1. his = Mowgli) (2. their = Wolf tribe)
4. (0. him = Mowgli)
5. (0. They = Bagheera/Mowgli)
6. (0. He = Mowgli)
7. (0. his = Colonel Hathi)
8. (0. his = Mowgli)
9. (0. himself = Baloo) (1. him = Mowgli)
10. (0. him = Mowgli) (their = Group of monkeys)
11. (0. he = Mowgli)
12. (0. he = Mowgli)
13.
14. (0. him = Baloo)

15. (0. him = Baloo) (1. his = Baloo)
16. (0. his = Colonel Hathi)
17. (0. himself = Shere Khan) (1. himself = Shere Khan)
18.
19.
20.
21.
22. (0. it = Nearby tree) (1. them = Flaming branches)
23.
24.
25. (0. His = Mowgli) (1. he = Mowgli)
26. (0. she = Beautiful Young Girl) (1. her = Beautiful Young Girl)
27. (0. it = Water pot) (1. her = Beautiful Young Girl) (2. her = Beautiful Young Girl)
28. (0. his = Mowgli)

A.2.5 Gold Standard Narrative Events

0. (Found in a basket - Bagheera/Mowgli) (Promptly takes wolf - Bagheera/Mowgli/Mother wolf)
1. (Raises along with own cubs and soon becomes well acquainted - Mother wolf/Mowgli) (Well acquainted with life - Mother wolf/Mowgli)
2. (Shown later - Mowgli) (Playing - Mowgli/Wolf siblings)
3. (Learns - Wolf tribe/Shere Khan) (Returned to the jungle - Shere Khan) (Realize - Wolf tribe/-Mowgli) (Taken to the Man Village - Wolf tribe/Mowgli)
4. (Volunteers to escort back - Bagheera/Mowgli)
5. (Leave very - Bagheera/Mowgli) (Determined to stay - Mowgli)
6. (Rest in a tree - Mowgli/Bagheera) (Tries to devour - Mowgli/Kaa) (Intervenes - Bagheera/-Mowgli/Kaa)
7. (Tries to join - Mowgli/Colonel Hathi/Winifred) (Led - Mowgli/Colonel Hathi/Winifred)
8. (Finds - Bagheera/Mowgli) (Fight - Bagheera/Mowgli) (Decides to leave - Mowgli)
9. (Soon meets up - Mowgli/Baloo) (Promises to raise - Mowgli/Baloo) (Never take to the Man Village back - Mowgli/Baloo)
10. (Kidnap - Group of monkeys/Mowgli) (Take to leader - Group of monkeys/Mowgli/King Louie)
11. (Offers to help - King Louie/Mowgli) (Stay in the jungle - King Louie/Mowgli) (Tell to make - King Louie/Mowgli)
12. (Was not raised humans - Mowgli) (Does not know - Mowgli) (Make fire - Mowgli)
13. (Arrive to rescue Bagheera/Baloo/Mowgli) (ensuing Bagheera/Baloo/Mowgli) (Palace is demolished to rubble - King Louie)
14. (Speaks - Bagheera/Baloo) (Convinces - Baloo/Mowgli/Shere Khan/Bagheera)
15. (Reluctantly explains - Baloo/Mowgli) (Accuses - Mowgli/Baloo) (Breaking promise -

Mowgli/Baloo) (Runs away - Mowgli)
16. (Sets in search of - Baloo/Mowgli) (Rallies the help - Bagheera/Colonel Hathi)
17. (Eavesdropping - Shere Khan/Bagheera/Colonel Hathi) (Now determined to hunt - Mowgli/Shere Khan) (Kill - Mowgli/Shere Khan)
18. (Encountered again - Mowgli/Kaa) (Escapes - Mowgli) (Thanks to the unwitting intervention - Shere Khan)
19. (Storm gathers - Mowgli/Vultures) (Encounters - Mowgli/Vultures) (Accept - Mowgli/Vultures)
20. (Appears shortly - Shere Khan) (Scaring off - Vultures/Shere Khan) (Confronting - Mowgli/Shere Khan)
21. (Rushes to the rescue - Baloo) (Tries to keep - Shere Khan/Mowgli/Baloo) (Injured - Baloo)
22. (Lightning strikes - Nearby Tree) (Swoop - Vultures/Shere Khan/Mowgli/Flaming branches) (Distract - Vultures/Shere Khan/Mowgli/Flaming branches) (Gathers - Vultures/Shere Khan/-Mowgli/Flaming branches) (Ties to tail - Mowgli/Shere Khan)
23. (Terrified of fire - Shere Khan) (Panics and runs off - Shere Khan)
24. (Take to the edge - Bagheera/Baloo/Mowgli) (Go there - Mowgli)
25. (Soon mind soon changes - Mowgli/Beautiful young girl) (Smitten from the village - Mowgli/Beautiful young girl) (Coming down by the riverside to fetch down - Mowgli/Beautiful young girl)
26. (Noticing - Mowgli/Beautiful young girl) (Accidentally drops - Beautiful young girl/Water pot)
27. (Retrieves - Mowgli/Water pot/Beautiful young girl) (follows in the Man-Village - Beautiful young girl/Mowgli)
28. (Chooses to stay - Mowgli) (Decided to head - Baloo/Bagheera)

Bibliography

[1] ADDIS, A., AND BORRAJO, D. From unstructured web knowledge to plan descriptions. In *Information Retrieval and Mining in Distributed Environments*. Springer, 2010, pp. 41–59.

[2] AINETO, D., JIMÉNEZ, S., AND ONAINDIA, E. Learning strips action models with classical planning. In *Twenty-Eighth International Conference on Automated Planning and Scheduling* (2018).

[3] AMATO, N. M., AND WU, Y. A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on* (1996), vol. 1, IEEE, pp. 113–120.

[4] AYLETT, R., DIAS, J., AND PAIVA, A. An affectively driven planner for synthetic characters. In *Icaps* (2006), pp. 2–10.

[5] BÄCKSTRÖM, C., AND NEBEL, B. Complexity results for sas+ planning. *Computational Intelligence 11*, 4 (1995), 625–655.

[6] BAMMAN, D., O'CONNOR, B., AND SMITH, N. A. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)* (2014), p. 352.

[7] BLUM, A. L., AND FURST, M. L. Fast planning through planning graph analysis. *Artificial intelligence 90*, 1-2 (1997), 281–300.

[8] BONNET, B., AND GEFFNER, H. Hsp: Heuristic search planner.

[9] BOTEA, A., ENZENBERGER, M., MÜLLER, M., AND SCHAEFFER, J. Macro-ff: Improving ai planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research 24* (2005), 581–621.

[10] BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., MALER, E., AND YERGEAU, F. Extensible markup language (xml). *World Wide Web Journal 2*, 4 (1997), 27–66.

[11] BYLANDER, T. Complexity results for planning. In *IJCAI* (1991), vol. 10, pp. 274–279.

[12] CAMBRIDGE DICTIONARY. Verbs: Types. `https://dictionary.cambridge.org/grammar/british-grammar/about-verbs/verbs-types`. [Online; accessed 26-November-2018].

[13] CAVAZZA, M., CHARLES, F., AND MEAD, S. J. Emergent situations in interactive storytelling. In *Proceedings of the 2002 ACM symposium on Applied computing* (2002), ACM, pp. 1080–1085.

[14] CHARLES, F., LOZANO, M., MEAD, S. J., BISQUERRA, A. F., AND CAVAZZA, M. Planning formalisms and authoring in interactive storytelling. In *Proceedings of TIDSE* (2003), vol. 3, p. 36.

[15] CLARK, K., AND MANNING, C. D. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2015), vol. 1, pp. 1405–1415.

[16] CLARK, K., AND MANNING, C. D. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667* (2016).

[17] CLARK, K., AND MANNING, C. D. Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323* (2016).

[18] CRESSWELL, S., AND GREGORY, P. Generalised domain model acquisition from action traces. In *ICAPS* (2011).

[19] CRESSWELL, S., MCCLUSKEY, T. L., AND WEST, M. Acquisition of object-centred domain models from planning examples. In *Nineteenth International Conference on Automated Planning and Scheduling* (2009).

[20] DE MARNEFFE, M.-C., AND MANNING, C. D. Stanford typed dependencies manual. Tech. rep., Technical report, Stanford University, 2008.

[21] DEEPTIMAHANTI, D. K., AND BABAR, M. A. An automated tool for generating uml models from natural language requirements. In *Proceedings of the 2009 IEEE/ACM international conference on automated software engineering* (2009), IEEE Computer Society, pp. 680–682.

[22] DODDINGTON, G. R., MITCHELL, A., PRZYBOCKI, M. A., RAMSHAW, L. A., STRASSEL, S., AND WEISCHEDEL, R. M. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC* (2004), vol. 2, p. 1.

[23] D'SOUZA, D. F., AND WILLS, A. C. *Objects, components, and frameworks with UML: the catalysis approach*, vol. 1. addison-Wesley Reading, 1998.

[24] EDELKAMP, S., AND HOFFMANN, J. Pddl2. 2: The language for the classical part of the 4th international planning competition. *4th International Planning Competition (IPC'04), at ICAPS'04* (2004).

[25] FIKES, R. E., AND NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence 2*, 3-4 (1971), 189–208.

[26] FINKEL, J. R., GRENAGER, T., AND MANNING, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd*

annual meeting on association for computational linguistics (2005), Association for Computational Linguistics, pp. 363–370.

[27] FINKEL, J. R., AND MANNING, C. D. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2009), Association for Computational Linguistics, pp. 326–334.

[28] FOX, M., AND LONG, D. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research 20* (2003), 61–124.

[29] GEREVINI, A., AND LONG, D. Plan constraints and preferences in pddl3. Tech. rep., Technical Report 2005-08-07, Department of Electronics for Automation, University of Brescia, Brescia, Italy, 2005.

[30] GOYAL, A., RILOFF, E., AND DAUMÉ III, H. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (2010), Association for Computational Linguistics, pp. 77–86.

[31] GREGORY, P., AND CRESSWELL, S. Domain model acquisition in the presence of static relations in the lop system. In *ICAPS* (2015), pp. 97–105.

[32] GREGORY, P., AND LINDSAY, A. Domain model acquisition in domains with action costs. In *ICAPS* (2016), pp. 149–157.

[33] GRISHMAN, R., AND SUNDHEIM, B. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics* (1996), vol. 1.

[34] HAYTON, T., GREGORY, P., LINDSAY, A., AND PORTEOUS, J. Best-fit action-cost domain model acquisition and its application to authorship in interactive narrative. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference* (2016).

[35] HELMERT, M. The fast downward planning system. *Journal of Artificial Intelligence Research 26* (2006), 191–246.

[36] HELMERT, M. Changes in pddl 3.1. *Unpublished summary from the IPC-2008 website* (2008).

[37] HOFFMANN, J. Analyzing search topology without running any search: On the connection between causal graphs and h+. *Journal of Artificial Intelligence Research 41* (2011), 155–229.

[38] HOFFMANN, J., AND NEBEL, B. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research 14* (2001), 253–302.

[39] HOFFMANN, J., PORTEOUS, J., AND SEBASTIA, L. Ordered landmarks in planning. *Journal of Artificial Intelligence Research 22* (2004), 215–278.

[40] KAUTZ, H., AND SELMAN, B. Unifying sat-based and graph-based planning. In *IJCAI* (1999), vol. 99, pp. 318–325.

[41] KAUTZ, H. A., SELMAN, B., ET AL. Planning as satisfiability. In *ECAI* (1992), vol. 92, Citeseer, pp. 359–363.

[42] KERTZ, L., KEHLER, A., AND ELMAN, J. Grammatical and coherence-based factors in pronoun interpretation. In *Proceedings of the 28th annual conference of the Cognitive Science Society* (2006), Lawrence Erlbaum Associates Mahwah, NJ, pp. 1605–1610.

[43] LIN, D., AND WU, X. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2* (2009), Association for Computational Linguistics, pp. 1030–1038.

[44] LINDSAY, A., READ, J., FERREIRA, J. F., HAYTON, T., PORTEOUS, J., AND GREGORY, P. Framer: Planning models from natural language action descriptions. In *Twenty-Seventh International Conference on Automated Planning and Scheduling* (2017).

[45] LOPER, E., AND BIRD, S. Nltk: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

[46] MAINPRICE, J., AND BERENSON, D. Human-robot collaborative manipulation planning using early prediction of human motion. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (2013), IEEE, pp. 299–306.

[47] MALMAUD, J., WAGNER, E., CHANG, N., AND MURPHY, K. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing* (2014), pp. 33–38.

[48] MANNING, C. D., SURDEANU, M., BAUER, J., FINKEL, J. R., BETHARD, S., AND MCCLOSKY, D. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)* (2014), pp. 55–60.

[49] MARCUS, M. P., MARCINKIEWICZ, M. A., AND SANTORINI, B. Building a large annotated corpus of english: The penn treebank. *Computational linguistics 19*, 2 (1993), 313–330.

[50] MARK O'BANNON. How to write a synopsis. `http://www.betterstorytelling.net/writingtools/howtowriteasynopsis.html`. [Online; accessed 19-October-2018].

[51] MATEAS, M., AND STERN, A. Structuring content in the façade interactive drama architecture. In *AIIDE* (2005), pp. 93–98.

[52] MCCLUSKEY, T., LIU, D., AND SIMPSON, R. M. Gipo ii: Htn planning in a tool-supported knowledge engineering environment. In *ICAPS* (2003), vol. 3, pp. 92–101.

[53] MCDERMOTT, D., GHALLAB, M., HOWE, A., KNOBLOCK, C., RAM, A., VELOSO, M., WELD, D., AND WILKINS, D. Pddl-the planning domain definition language.

[54] MCDERMOTT, D. M. The 1998 ai planning systems competition. *AI magazine 21*, 2 (2000), 35.

[55] MILLER, G. A. Wordnet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[56] MUISE, C. Planning. domains. *ICAPS system demonstration* (2016).

[57] MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE 77*, 4 (1989), 541–580.

[58] NAVIGLI, R. Word sense disambiguation: A survey. *ACM computing surveys (CSUR) 41*, 2 (2009), 10.

[59] NEWELL, A., AND SIMON, H. A. Gps, a program that simulates human thought. Tech. rep., RAND CORP SANTA MONICA CALIF, 1961.

[60] OPENNLP, A. Apache software foundation. *URL http://opennlp. apache. org* (2011).

[61] ORKIN, J., AND ROY, D. Understanding speech in interactive narratives with crowd-sourced data. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference* (2012).

[62] OXFORD DICTIONARIES. Types of noun. `https://en.oxforddictionaries.com/grammar/types-of-noun`. [Online; accessed 25-September-2018].

[63] PENBERTHY, J. S., WELD, D. S., ET AL. Ucpop: A sound, complete, partial order planner for adl. *Kr 92* (1992), 103–114.

[64] PIACENZA, A., GUERRINI, F., ADAMI, N., LEONARDI, R., TEUTENBERG, J., PORTEOUS, J., AND CAVAZZA, M. Changing video arrangement for constructing alternative stories. In *Proceedings of the 19th ACM international conference on Multimedia* (2011), ACM, pp. 811–812.

[65] POIBEAU, T., AND KOSSEIM, L. Proper name extraction from non-journalistic texts. *Language and computers 37* (2001), 144–157.

[66] PORTEOUS, J., AND CAVAZZA, M. Controlling narrative generation with planning trajectories: the role of constraints. In *Joint International Conference on Interactive Digital Storytelling* (2009), Springer, pp. 234–245.

[67] PORTEOUS, J., CAVAZZA, M., AND CHARLES, F. Narrative generation through characters' point of view. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1* (2010), International Foundation for Autonomous Agents and Multiagent Systems, pp. 1297–1304.

[68] PORTEOUS, J., CHARLES, F., AND CAVAZZA, M. Networking: using character relationships for interactive narrative generation. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (2013), International Foundation for Autonomous Agents and Multiagent Systems, pp. 595–602.

[69] PORTEOUS, J., TEUTENBERG, J., PIZZI, D., AND CAVAZZA, M. Visual programming of plan dynamics using constraints and landmarks. In *ICAPS* (2011).

[70] PRADHAN, S., MOSCHITTI, A., XUE, N., URYUPINA, O., AND ZHANG, Y. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task* (2012), Association for Computational Linguistics, pp. 1–40.

[71] PRADHAN, S., RAMSHAW, L., MARCUS, M., PALMER, M., WEISCHEDEL, R., AND XUE, N. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task* (2011), Association for Computational Linguistics, pp. 1–27.

[72] PROPP, V. *Morphology of the Folktale*, vol. 9. University of Texas Press, 2010.

[73] RAGHUNATHAN, K., LEE, H., RANGARAJAN, S., CHAMBERS, N., SURDEANU, M., JURAFSKY, D., AND MANNING, C. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (2010), Association for Computational Linguistics, pp. 492–501.

[74] RATINOV, L., AND ROTH, D. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (2009), Association for Computational Linguistics, pp. 147–155.

[75] RIEDL, M. O., AND YOUNG, R. M. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research 39*, 1 (2010), 217–268.

[76] RITTER, A., CLARK, S., ETZIONI, O., ET AL. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing* (2011), Association for Computational Linguistics, pp. 1524–1534.

[77] SCHEUTZ, M., KRAUSE, E., OOSTERVELD, B., FRASCA, T., AND PLATT, R. Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (2017), International Foundation for Autonomous Agents and Multiagent Systems, pp. 1378–1386.

[78] SCOOBYDOO.WIKIA. What a night for a knight. `http://scoobydoo.wikia.com/wiki/What_a_Night_for_a_Knight`.

[79] SIMPSON, R. Structural domain definition using gipo iv. *Proceedings of the Second International Competition on Knowledge Engineering for Planning and Scheduling* (2007).

[80] SIMPSON, R. M., KITCHIN, D. E., AND MCCLUSKEY, T. Planning domain definition using gipo. *The Knowledge Engineering Review 22*, 02 (2007), 117–134.

[81] SIMPSON, R. M., MCCLUSKEY, T. L., ZHAO, W., AYLETT, R. S., AND DONIAT, C. Gipo: an integrated graphical tool to support knowledge engineering in ai planning. In *Sixth European Conference on Planning* (2014).

[82] STANFORD CORENLP. Corefannotator. `https://stanfordnlp.github.io/CoreNLP/coref.html`.

[83] THOMASON, J., ZHANG, S., MOONEY, R. J., AND STONE, P. Learning to interpret natural language commands through human-robot dialog. In *IJCAI* (2015), pp. 1923–1929.

[84] TJONG KIM SANG, E. F., AND DE MEULDER, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (2003), Association for Computational Linguistics, pp. 142–147.

[85] VALLS-VARGAS, J., ONTANÓN, S., AND ZHU, J. Toward character role assignment for natural language stories. In *Proceedings of the Ninth Artificial Intelligence and Interactive Digital Entertainment Conference* (2013), pp. 101–104.

[86] VAQUERO, T. S., ROMERO, V., TONIDANDEL, F., AND SILVA, J. R. itsimple 2.0: An integrated tool for designing planning domains. In *ICAPS* (2007), pp. 336–343.

[87] VAQUERO, T. S., SILVA, J. R., FERREIRA, M., TONIDANDEL, F., AND BECK, J. C. From requirements and analysis to pddl in itsimple3. 0. *Proceedings of the Third International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2009* (2009), 54–61.

[88] VAQUERO, T. S., TONACO, R., COSTA, G., TONIDANDEL, F., SILVA, J. R., AND BECK, J. C. itsimple4. 0: Enhancing the modeling experience of planning problems. In *System Demonstration–Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12)* (2012), pp. 11–14.

[89] VAQUERO, T. S., TONIDANDEL, F., AND SILVA, J. R. The itsimple tool for modeling planning domains. *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, Califormia, USA* (2005).

[90] W3TECHS. Usage of content languages for websites. `https://w3techs.com/technologies/overview/content_language/all`.

[91] WIKIPEDIA. The jungle book (1967 film). `https://en.wikipedia.org/wiki/The_Jungle_Book_(1967_film)`.

[92] YORDANOVA, K. From textual instructions to sensor-based recognition of user behaviour. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces* (2016), ACM, pp. 67–73.

[93] YOUNG, R. M. Notes on the use of plan structures in the creation of interactive plot. In *AAAI Fall Symposium on Narrative Intelligence* (1999), pp. 164–167.

[94] ZHAI, H., LINGREN, T., DELEGER, L., LI, Q., KAISER, M., STOUTENBOROUGH, L., AND SOLTI, I. Web 2.0-based crowdsourcing for high-quality gold standard development in clinical natural language processing. *Journal of medical Internet research 15*, 4 (2013).

[95] ZHOU, G., AND SU, J. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (2002), Association for Computational Linguistics, pp. 473–480.